1. Einleitung

Wir danken Ihnen für den Erwerb unserer Produkte. Damit steht Ihnen der weltweite Kundensupport für amtec-Produkte zur Verfügung. Unsere Mitarbeiter und Partner möchten Ihnen helfen, die gewünschten Resultate schnell und professionell zu erzielen.
Das Verzeichnis der Vertriebs- und Kundendienstpartner der amtec GmbH befindet sich im Anhang dieses Handbuchs.

1.1. Copyright-Informationen

© 1991-1999 amtec GmbH Germany
Alle Rechte vorbehalten. Die Reproduktion, Anpassung oder Übersetzung ist ohne vorherige schriftliche Einwilligung außer im Rahmen der Urheberrechtsgesetze verboten.

1.2. Garantieerklärung zur Dokumentation

Die in diesem Dokument enthaltenen Informationen können ohne vorherige Ankündigung verändert werden. Die amtec GmbH gibt bezüglich des Materials keine Garantie. Dies umfaßt auch - beschränkt sich jedoch nicht auf - die gesetzlichen Gewährleistungen, daß die Waren für den normalen Gebrauch bzw. für den vorbestimmten Zweck geeignet sind.
Die amtec GmbH haftet weder für Fehler in diesem Handbuch, noch für beiläufig entstandene Schäden oder Folgeschäden, die in Verbindung mit der Bereitstellung oder Verwendung dieses Handbuchs entstehen.

Warenzeichen

PowerCube™ ist ein eingetragenes Warenzeichen der amtec GmbH. Beachten Sie die Bedingungen zur Verwendung des PowerCube™-Logos in der Einleitung dieses Handbuchs.
Windows™ ist ein eingetragenes Warenzeichen der Microsoft Corporation.

Versionshinweis

Diese Dokumentation bezieht sich auf die PowerCube™-Produktfamilie ab Version 3.5.00.
Die Versionsnummer kann folgenderweise aufgeschlüsselt werden:
3: bezieht sich auf die aktuelle Hardware-Version der PowerCube™-Module. Bei Änderungen an dieser Stelle wenden Sie sich bitte an Ihren Service-Partner.
5: bezieht sich auf die Software-Version der aktuellen API-Schnittstelle *m* x*dll* (x = 5) sowie der optional angebotenen Software-Bibliotheken. Bei Änderungen an dieser Stelle wenden Sie sich bitte an Ihren Service-Partner.
00: bezeichnen Unterreleases. Neue Versionen können von Ihnen installiert werden.

1.3. Software-Lizenzvertrag

Die Verwendung der PowerCube™ Software unterliegt den nachstehend aufgeführten Bestimmungen des amtec Softwarelizenzvertrags. Indem Sie die Software installieren, erkennen Sie diese Vertragsbedingungen vollständig an.
Sofern kein gesonderter schriftlicher Vertrag mit amtec abgeschlossen wurde, unterliegt die Verwendung der mitgelieferten Software den folgenden Bestimmungen:

Lizenzerteilung

amtec gewährt Ihnen das Recht, eine (1) Kopie der Software zu verwenden. "Verwenden" bedeutet in diesem Fall, die Software zu speichern, zu laden, zu installieren, auszuführen oder anzuzeigen. Der Lizenznehmer darf die Software weder verändern noch irgendwelche Lizensierungs- bzw. Steuerungsfunktionen der Software deaktivieren. Ist die Software für "gleichzeitige Nutzung" lizensiert, dürfen nicht mehr

Benutzer die Software zur gleichen Zeit verwenden als maximal dazu ermächtigt.

Eigentumsrecht Die Software ist Eigentum von amtec und durch sie urheberrechtlich geschützt. Die Lizenz beinhaltet kein Eigentumsrecht an der Software und stellt keineswegs einen Verkauf von irgendwelchen Rechten daran dar. Drittanbieter, mit denen amtec zusammenarbeitet, sind dazu befugt, im Falle einer Verletzung dieser Lizenzbedingungen juristische Schritte zum Schutz der Rechte einzuleiten.

Kopien & Anpassungen Kopien und Anpassungen der Software dürfen ausschließlich zum Zwecke der Archivierung oder im Rahmen der autorisierten Verwendung der Software vorgenommen werden. Dabei müssen auf alle Kopien und Anpassungen alle in der Original-Software enthaltenen Copyright-Hinweise mit kopiert werden. Die Software darf nicht in ein öffentlich zugängliches System kopiert werden, sofern dies nicht ausdrücklich gestattet wurde.
De-Assemblieren oder Entschlüsseln untersagt. Die Software darf keinesfalls ohne vorherige schriftliche Genehmigung von amtec de-assembliert oder de-kompiliert werden. Die Entschlüsselung der Software ist untersagt, es sei denn, sie ist im Rahmen des Betreibens der Software erforderlich.

Übertragung Ihre Lizenz erlischt automatisch, sobald die Software in irgendeiner Weise übertragen wird. Bei der Eigentumsübertragung müssen Sie dem Empfänger die gesamte Software einschließlich aller Kopien und der zugehörigen Dokumentation überlassen. Der Empfänger muß diese Lizenzbedingungen als Bedingung der Eigentumsübertragung anerkennen.

Vertragsbeendigung amtec hat das Recht, dem Lizenznehmer sofort die Lizenz zu entziehen, wenn dieser eine der Bedingungen dieses Vertrages verletzt. Bei vorzeitiger Beendigung des Vertrages ist der Lizenznehmer verpflichtet, die Software einschließlich der zugehörigen Dokumentation und aller Kopien oder Änderungen in jeglicher Form zu zerstören.

Exportbestimmungen Es ist untersagt, die Software bzw. eine Kopie oder Anpassung zu exportieren oder zu reexportieren, wenn dadurch anwendbares Recht oder Rechtsvorschriften verletzt werden

1.4. Bestimmungen zur Verwendung des PowerCubeTM-Logos

Als Anwender der PowerCube™ Produktfamilie steht es Ihnen frei, das PowerCube™-Logo für Ihre Dokumentationen und Präsentationen zu verwenden. Ein konsistentes "Look and Feel" ist dabei Voraussetzung für die Wiedererkennung und den Erfolg Ihrer Publikation. Wir bitten Sie jedoch aus rechtlichen Gründen, vor Herausgabe der Publikation, ein Belegexemplar an amtec zu schicken.
Beachten Sie dabei folgende Richtlinien:
Das PowerCube™-Logo muß schwarz dargestellt werden. Diese Darstellung erfolgt stets auf hellem Hintergrund.
Das PowerCube™-Logo sollte auf Drucksachen nie kleiner als 20 mm dargestellt werden

Ränder In einem definierten Bereich um das Logo herum dürfen sich keine Schriftzüge oder Logos anderer Unternehmen befinden. Der einzuhaltende Freiraum rechts und links des Logos sollte jeweils 5% der Logobreite nicht unterschreiten. Der einzuhaltende Freiraum oberhalb und unterhalb des Logos sollte jeweils 25% der Logohöhe nicht unterschreiten.

Logo-Integrität Änderungen am PowerCube™-Logo sind nicht zulässig. Um die Integrität des Logos aufrecht zu halten, verwenden Sie bitte stets das Original-Logo in elektronischer Form.

Logo im Web Verwenden Sie stets die Darstellung in schwarz auf weißem Hintergrund.

Warenzeichen In allen Veröffentlichungen ist darauf hinzuweisen, daß PowerCube™ ein eingetragenes Warenzeichen der amtecamtec GmbH ist.

1.5. Abkürzungen und Begriffserläuterungen

CPU	**C**entral **P**rocessing **U**nit
RISC	**R**educed **I**nstruction **S**et **C**ode
Tool-Center-Point	Der Tool-Center-Point ist der Punkt am Manipulator, auf den sich alle Bewegungen beziehen. Meistens ist dieser Punkt die Spitze eines Werkzeuges, das an den Manipulator montiert ist.
Weltkoordinaten	Die Weltkoordinaten (x) beschreiben ein rechtwinkliges Koordinatensystem mit den Achsen X, Y, Z und den Winkeln a, b, c. Die Winkel a, b, c beschreiben die Drehung um die Achsen X, Y, Z. Die Weltkoordinaten können in Roboterkoordinaten umgerechnet werden, wobei es mehrere Lösungen geben kann. Sie werden häufig auch durch den Begriff „Orientierung" bezeichnet.
Roboterkoordinaten	Roboterkoordinaten (q) sind die Koordinaten der Gelenke. Die Roboterkoordinaten können in eindeutige Weltkoordinaten umgerechnet werden.
Freiheitsgrad	Ein Freiheitsgrad beschreibt die Bewegung entlang einer Achse oder die Bewegung um eine Achse. Im Weltkoordinatensystem gibt es 6 Freiheitsgrade.
Manipulator	Ein Manipulator (Roboter) beschreibt die Summe aller Module. Jedes Modul ist verantwortlich für die Bewegung in oder um einen Freiheitsgrad. Die Bewegungen eines Manipulators dienen der Interaktion mit seiner Umwelt.
Punkt-zu-Punkt (PTP)	Eine Punkt-zu-Punkt Bewegung ist die Bewegung des Tool-Center-Points von Punkt A nach Punkt B, wobei die Bewegung der einzelnen Module nicht synchronisiert wird. Somit ist die Bahn des TCP nicht festgelegt.
Bahnsteuerung	Als Bahnsteuerung bezeichnet man eine synchronisierte Bewegung von Punkt A nach Punkt B, entlang eines vorgegebenen Weges.

2. Sicherheitshinweise

Bei der Verwendung der PowerCube™-Module und deren Kombination müssen alle grundlegenden Sicherheitsvorschriften immer befolgt werden. Die Module, die Zubehörteile und Zusatzeinrichtungen dürfen erst nach Kenntnisnahme der Betriebsanleitung und nur durch hierfür unterwiesene Personen montiert und in Betrieb genommen werden.
Es sind nur die vom Hersteller empfohlenen oder in der Betriebsanleitung enthaltenen Zusatzeinrichtungen zu verwenden.

2.1. Sorgfaltspflicht des Betreibers

Die PowerCube™-Module werden unter Berücksichtigung einer Gefährdungsanalyse und nach sorgfältiger Auswahl der einzuhaltenden harmonisierten Normen sowie weiterer technischer Spezifikationen konstruiert und gebaut. Sie entsprechen damit dem Stand der Technik und ermöglichen ein Höchstmaß an Sicherheit während des Betriebes.

Konstruktive Veränderungen an den PowerCube™-Modulen dürfen nur nach schriftlicher Genehmigung durch den Hersteller vorgenommen werden! Zuwiderhandlungen schließen jede Gewährleistung durch den Hersteller aus!
Die Sicherheit der PowerCube™-Module kann in der betrieblichen Praxis nur dann umgesetzt werden, wenn alle dafür erforderlichen Maßnahmen getroffen werden. Es unterliegt der Sorgfaltspflicht des Betreibers der PowerCube™-Module diese Maßnahmen zu planen und ihre Ausführung zu kontrollieren.
Der Betreiber muß insbesondere sicherstellen, daß

- die PowerCube™-Module nur bestimmungsgemäß genutzt werden (siehe Kapitel Produktvarianten).
- die PowerCube™-Module nur in einwandfreiem, funktionstüchtigem Zustand betrieben werden und besonders die Sicherheitseinrichtungen regelmäßig auf ihre Funktionstüchtigkeit überprüft werden.
- erforderliche persönliche Schutzausrüstungen für Bedienungs-, Wartungs- und Reparaturpersonal zur Verfügung stehen und getragen werden.
- die Betriebsanleitung stets in einem leserlichen Zustand und vollständig am Einsatzort der PowerCube™-Module zur Verfügung steht.
- nur dafür qualifiziertes und autorisiertes Personal die PowerCube™-Module bedient, wartet und repariert. Bei der Wartung und Reparatur ist darauf zu achten, daß nur Originalteile verwendet werden.
- das Personal regelmäßig in allen zutreffenden Fragen von Arbeitssicherheit und Umweltschutz unterwiesen wird sowie die Betriebsanleitung und insbesondere die darin enthaltenen Sicherheitshinweise kennt.
- alle an den PowerCube™-Modulen selbst angebrachten Sicherheits- und Warnhinweise leserlich sind und nicht entfernt werden.

2.2. Sicherheitsmaßnahmen bei Normalbetrieb

Die PowerCube™-Module dürfen nur von befugten Personen bedient werden, denen die Betriebsanleitung bekannt ist und die danach arbeiten können.
Vor dem Einschalten der PowerCube™-Module ist zu überprüfen und sicherzustellen, daß

- sich keine Personen im Arbeitsbereich der PowerCube™-Module aufhalten.
- niemand durch das Anlaufen der PowerCube™-Module verletzt

werden kann.
- die Stromversorgung der in der Betriebsanleitung spezifizierten Stromversorgung entspricht.

Vor jedem Produktionsbeginn sind die PowerCube™-Module, Zuleitungen und Zubehör auf sichtbare Schäden zu überprüfen und es ist sicherzustellen, daß sie nur in einwandfreiem Zustand betrieben werden! Festgestellte Mängel sind sofort zu melden.
Vor jedem Produktionsbeginn sind Material und Gegenstände aus dem Arbeitsbereich der PowerCube™-Module zu entfernen, die nicht für die Produktion erforderlich sind.
Vor jedem Produktionsbeginn ist zu prüfen und sicherzustellen, daß alle Sicherheitseinrichtungen einwandfrei funktionieren.
Vor der Montage, Wartung und Einrichtung von Zusatzeinrichtungen und Zubehörteilen sind die PowerCube™-Module und deren Kombination netzfrei zu schalten.

2.3. Bestimmungsgemäße Verwendung

Es ist sicherzustellen, daß die Module im Betrieb nicht von den Spezifikationen (Nutzlast, Strombedarf u. ä.) abweichen, die in den betreffenden Datenblättern enthalten sind.

2.4. Unzulässige Änderungen

Die PowerCube™-Module dürfen kundenseitig nur von Personal geöffnet werden, das durch eine spezielle Schulung von amtec dafür autorisiert wurde (Ausnahmen s. u. Wartungshinweise). Es ist darauf zu achten, daß die Module keine Lasten bewegen, für die sie konstruktionsbedingt nicht ausgelegt sind. Die Module sind werksseitig bei Auslieferung nur für den Betrieb an einem bestimmten Bussystem geeignet (RS232, CAN-Bus oder Profi-Bus). Dies bedeutet, daß die PowerCube™-Module nicht ohne Konfigurationsänderungen an einer anderen Schnittstelle betrieben werden können! Diese Umstellung setzt das Öffnen der Module voraus, so daß eine Neukonfigurierung der Schnittstelle nur von amtec vorgenommen werden darf.

2.5. Sicherheitsmaßnahmen bei Wartung und Instandhaltung

Die in der Betriebsanleitung vorgeschriebenen Inspektions- und Wartungsintervalle sind Mindestforderungen und sollten nicht größer sein als die sonst im Anwendungsbereich üblichen.
Die Wartungs- und Reparaturanleitungen zu den Einzelkomponenten in dieser Betriebsanleitung sind zu beachten.
Vor Wartungs- und Reparaturarbeiten ist der Hauptschalter für die Stromversorgung auszuschalten und mit einem Vorhängeschloß zu sichern. Der Schlüssel zu diesem Schloß muß in Händen der Person sein, die die Wartungs- oder Reparaturarbeit ausführt.
Beim Austausch schwerer Maschinenteile sind nur geeignete und einwandfreie Lastaufnahmeeinrichtungen und Anschlagmittel zu verwenden.
Vor Wartungs- und Reparaturarbeiten ist sicherzustellen, daß alle eventuell zu berührenden Teile der PowerCube™-Module sich auf Raumtemperatur abgekühlt haben.
Vor der Ausführung von Wartungs- oder Reparaturarbeiten ist der Zugang zum Arbeitsbereich der PowerCube™-Module für unbefugte Personen zu sperren! Ein Hinweisschild ist anzubringen oder aufzustellen, das auf die Wartungs- oder Reparaturarbeit aufmerksam macht.
Umweltgefährdende Schmier-, Kühl- oder Reinigungsmittel sind ordnungsgemäß zu entsorgen.
Vor Inbetriebnahme nach Wartungs- oder Reparaturarbeiten

- sind gelöste Schraubverbindungen auf festen Sitz zu prüfen.
- ist sicherzustellen, daß vorher entfernte Behälterdeckel, Siebe oder Filter wieder eingebaut sind.

Nach Abschluß von Wartungs- oder Reparaturarbeiten und vor der Wiederaufnahme der Produktion ist sicherzustellen, daß

- alle für die Ausführung der Wartungs- oder Reparaturarbeiten benötigten Materialien, Werkzeuge und sonstige Ausrüsten aus dem Arbeitsbereich der Anlage entfernt wurden.
- eventuell ausgetretene Flüssigkeiten entfernt wurden.
- alle Sicherheitseinrichtungen der Anlage einwandfrei funktionieren.

2.6. Arbeiten an der elektrischen Ausrüstung

Reparaturarbeiten an elektrischen Ausrüstungen der PowerCube™-Module dürfen nur von einer ausgebildeten Elektrofachkraft ausgeführt werden.
Elektrische Ausrüstungen sind regelmäßig zu überprüfen.
Lose Verbindungen müssen wieder befestigt werden.
Beschädigte Leitungen und Kabel sind sofort auszutauschen.
Der Schaltschrank ist stets geschossen zu halten! Zugang ist nur befugten Personen erlaubt.
Schaltschränke und andere Gehäuse von elektrischen Ausrüstungen sind zur Reinigung niemals mit einem Wasserschlauch abzuspritzen.

2.7. Umweltschutzvorschriften beachten

Bei allen Arbeiten an und mit den PowerCube™-Modulen sind die gesetzlichen Pflichten zur Abfallvermeidung und ordnungsgemäßen Verwertung/Beseitigung von Abfallprodukten einzuhalten.
Insbesondere bei Installations-, Reparatur- und Wartungsarbeiten dürfen wassergefährdende Stoffe wie

- Schmierfette und -öle
- Hydrauliköle
- Kühlmittel
- lösungsmittelhaltige Reinigungsflüssigkeiten

nicht den Boden belasten oder in die Kanalisation gelangen.
Diese Stoffe müssen in geeigneten Behältern aufgefangen, aufbewahrt, transportiert und entsorgt werden.

2.8. Haftung

Die Haftung des Herstellers für Mängel und daraus entstehende Folgen wird ausgeschlossen, wenn die Mängel verursacht sind durch:

- nicht bestimmungsgemäße Verwendung der PowerCube™-Module,
- fehlerhafte Montage durch den Betreiber,
- fehlerhafte oder nachlässige Wartung,
- Verwendung ungeeigneter Schmierstoffe,
- natürlichen Verschleiß,
- weitere Verwendung nach Auftreten von Funktionsstörungen,
- weitere Verwendung nach erkannten oder vermutbaren Beschädigungen an Material und Zubehör,
- nicht schriftlich vom Hersteller genehmigte Eingriffe an PowerCube™-Modulen und deren Zubehör durch den Betreiber.

3. Aufbau und Funktionsweise

In den folgenden Kapiteln erhalten Sie detaillierte Informationen über den internen Aufbau und die Funktionsweise der PowerCube™-Produktlinie. Sie erfahren, welche Produktvarianten existieren und welche Schnittstellen das System bietet.

3.1. Bestellbezeichnung der PowerCube™-Module

Bitte beachten Sie, daß die Bestellbezeichnung der PowerCube™-Module auf dem Lieferschein in bestimmten Fällen aus internen abwicklungstechnischen Gründen nicht mit der Bezeichnung auf der Verpackung bzw. auf den PowerCube™-Modulen übereinstimmen.
Die nachfolgende Zusammenstellung gibt Ihnen eine Übersicht über alle Module (inklusive Optionen) und die dazugehörigen Bestellnummern.

3.2. Funktionsprinzipien

Das PowerCube™-Produktsystem integriert alle Elemente zur Motorsteuerung in den Antrieb selbst. Ein PowerCube™ verbindet folgende Komponenten auf kleinstem Raum:

- einen hochdynamischen elektronisch kommutierten Gleichstrommotor, in den meisten Modulen mit einstufigem Getriebe.
- einen extrem leistungsfähigen Torque Controller, der den Motor kommutiert.
- einen optischen Encoder (500 Strich auf 2000 Flanken hochgerechnet).
- einen hochpräzisen Motion Controller für die Bewegungssteuerung auf der Basis von Encoderfeedback.

Überwachungselektronik für:

- Motorstrom
- Temperatur der Motorwicklung
- Temperatur der Endstufentransistoren.
- eine serielle High-Speed Industriebusschnittstelle zur Vernetzung von bis zu 31 PowerCube™-Antrieben.
- ein I/O-Interface für zusätzliche Ein-/Ausgänge sowie 24 V-Stromversorgung im Notaus-Regime.

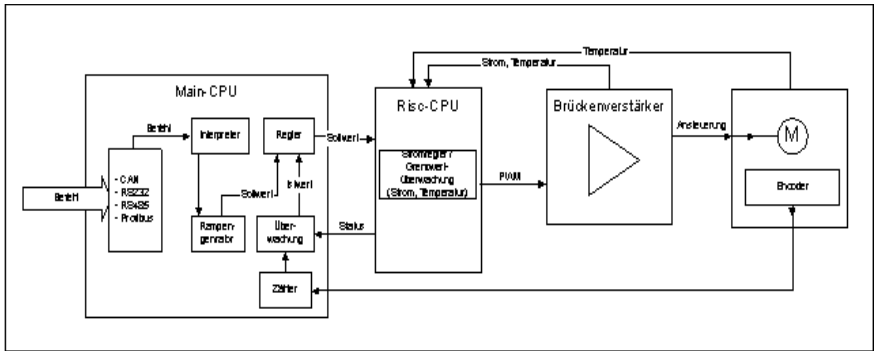
Das I/O-Interface kann in drei verschiedenen Konfigurationen geliefert werden:

Typ I:	4 IN / 4 OUT
Typ II:	Encoder-Signal-Ausgang
Typ III:	Ausgang für externes Encoder-Signal

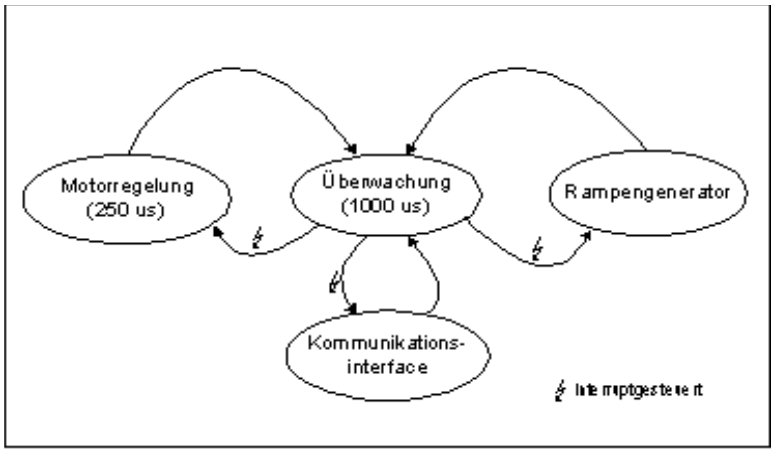
Die verwendeten Motoren sind bürstenlose Gleichstrommotoren. Der Vorteil dieser Motoren liegt in der beliebigen Positioniermöglichkeit und in dem geringen Verschleiß.

3.3. Funktionsweise

Das folgende Schaubild zeigt in vereinfachter Form die Funktionsweise der PowerCube™-Module .



Im Hauptprozessor der PowerCube™-Module laufen vier Regel- und Steuerkreise (Tasks).



Motorregelung

Die Hauptaufgabe des Mikroprozessors liegt in der Regelung des Motors. Dieser Regelkreis arbeitet mit einer Periode von 250 µs. Das stark vereinfachte Schaubild soll Ihnen einen Einblick in die Funktionsweise des Regelkreises verschaffen.

Überwachung

Die Überwachungsschleife wiederholt sich alle 1000 µs (1ms) und überprüft dabei die folgenden Werte:

- Endlage
- Überstrom
- Unterspannung
- Hallgeber
- Temperatur Motor
- Temperatur Endstufe
- Schleppfehler.

Der Betriebszustand der PowerCube™-Module kann über das Statuswort CubeState abgefragt werden. Dort werden auch die Reaktionen des Betriebssystems auf eventuelle Fehlerzustände beschrieben.

Kommunikation

Das Interface arbeitet asynchron, d. h. auf Anfrage, und muß dabei die folgenden Punkte erfüllen:

- Interpretation der Anfrage
- Ausführung der Anfrage
- Antworten

Rampengenerator

Bei Bewegung der Module besteht die Aufgabe darin, einen bestimmten Weg nach vorgegebenen Parametern zurückzulegen. Diese Aufgabe läßt sich in drei Bereiche einteilen. Zunächst erfolgt eine konstante Beschleunigung, die in eine konstante Geschwindigkeit übergeht und in einer konstanten negativen Beschleunigung (bremsen) endet. Die Berechnung dieser Rampe erfolgt im Interpolator. Dieser modulinterne Interpolator berechnet unter Berücksichtigung des bisherigen

Rampenverlaufs alle 250 µs eine neue Sollposition für den Lageregler.

3.4. Encoder

An der Achswelle in den Modulen befindet sich ein inkrementaler Encoder (Lochscheibe, Lochmaske). Auf der Lochscheibe befinden sich 500 Schlitze und ein Nullpunktschlitz. Beide Scheiben werden von drei Lichtquellen durchleuchtet. Die Lichtimpulse werden von drei Detektoren in elektrische Impulse umgesetzt. Mit Hilfe der elektrischen Impulse ist nun eine genaue Positionsbestimmung möglich. Um einen Bezug zum umgebenden Koordinatensystem zu erhalten, muß das Modul eine Referenzfahrt (Homing) durchführen.

3.5. Referenzfahrt (Homing)

Vor dem ersten Fahrkommando muß das Antriebsmodul referenziert werden, das heißt es muß einen Referenzpunkt anfahren, um einen absoluten Bezug zum umgebenden Koordinatensystem zu erhalten. Die Referenzfahrt wird mit dem Kommando >Home< ausgelöst. Wird dieses Kommando aufgerufen, setzt sich das Modul in Richtung des Referenzschalters in Bewegung. Zunächst wird die im Modul eingestellte Flanke des Referenzschalters gesucht. Anschließend wird der Nullpunkt des Encoders angefahren. Hier stoppt das Modul die Fahrt und meldet die für den Referenzpunkt voreingestellte Position.

3.6. Elektrische Schnittstellen der PowerCube™ Module

Die PowerCube™-Produktfamilie beinhaltet verschiedene Varianten von Antriebsmodulen, deren interner Aufbau gleichartig ist. Bitte wenden Sie sich an Ihren lokalen PowerCube-Distributor, um Auskunft über die technischen Daten der Produkte zu erhalten.
Alle PowerCube™ Module verfügen über zwei Steckverbinder, die zum Datenaustausch und zur 24V-Spannungsversorgung dienen. Hier handelt es sich um einen 7-poligen Stecker mit Mischkontaktierung (Input) und eine 7-polige Buchse mit Mischkontaktierung (Output). Die Module der Baugröße 110 (500 W) verfügen zusätzlich über zwei 3-polige Steckverbinder (Input/Output) zur 48V-Spannungsversorgung. Mit Ausnahme der Handgelenk(PW)- und Greifer(PG)-Module verfügen alle Module über einen weiteren 15-poligen Steckverbinder für zusätzliche Ein/Ausgänge. Zur Pin-Belegung der Steckverbinder siehe weiter unten.

Bus	CAN	Profibus-DP	RS232
Norm	ISO 118 98	DIN 19245 EN 50 170 EN 50254	EIA 232
Übertragungsrate	50 KBit/s - 1MBit/s	9,6KBit/s - 12MBit/s	1,2 - 115,2 KBit/s
Nutzdaten	max. 8 Byte	8 Ausgang 16 Eingang	max. 8 Byte
Max. Länge	1km (50 KBit/s)	1km (93 KBit/s)	15m (19,2 KBit/s)
Kommunikation	-	-	vollduplex
Signale	differentiell	differentiell	single-ended
Ausgangsstrom	-	-	500 mA

Tabelle 1

Ausgangsspannung	-	-	+/- 25 V (max) +/- 5 V (min)
Input Level min.	-	-	+/- 3 V
Abschlußwider-stand (Ohm)	120 Ohm	120 Ohm	0 Ohm
Datensicherung	CRC (16 Bit)	FCS	-
Teilnehmer	max .31	max. 31	max. 8

Tabelle 1

Sie können die Module mit einem von drei Kommunikationsprotokollen betreiben : CAN, Profibus-DP oder RS232. Die Belegung der Steckverbinder ist von der intern im PowerCube™ eingestellten Konfiguration abhängig. Das voreingestellte Kommunikationsprotokoll kann auf Wunsch geändert werden ohne die Module oder die Verbindungskabel austauschen zu müssen. Wenden Sie sich in diesem Fall bitte an Ihren Service-Partner.

Pinning der Input/Output-Verbinder (7-polig)

Pin	CAN	Profibus-DP	RS232
A1	24 V	24 V	24 V
A2	0 V	0 V	0 V
1	CAN-Hi	B	-
3	CAN-Lo	A	-
4	Schirm	Schirm	Schirm
2	-	-	back
5	-	-	In/Out

Tabelle 2

Pinning des zusätzlichen 15-poligen I/O-Interfaces

Pin	Typ I 4 I/O	Typ II Encoder Output	Typ III External Encoder In
11	GND extern	GND extern	-
4	-	GND extern	GND extern
12	Input 3	Z1	Z1
5	-	/Z1	/Z1
13	Output 0	Z2	Z2
6	Output 1	/Z2	/Z2
14	Output 2	N	N
7	Output 3	/N	/N
1	+24 V extern	+ 24 V extern	+ 24 V extern
9	Input 0 (SWR)	Input 0 (SWR)	Input 0 (SWR)
2	Input 1(SW1)	Input 1(SW1)	Input 1(SW1)
10	Input 2 (SW2)	Input 2 (SW2)	Input 2 (SW2)

Tabelle 3

3	GND	GND	GND
15	+ 24 V (Output)	-	-
8	GND extern	GND extern	GND extern

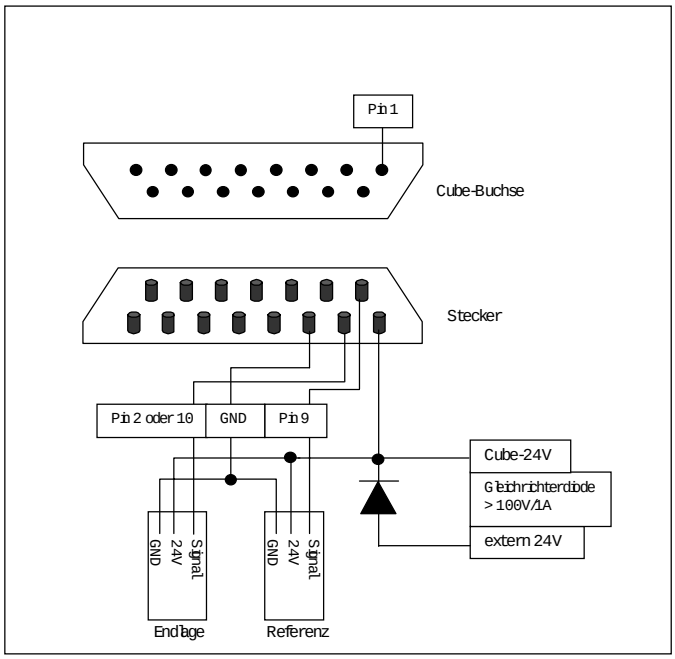
Tabelle 3

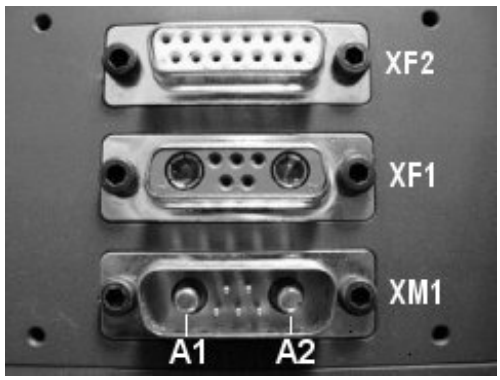
Es gibt zwei Modi der Stromversorgung für die PowerCube™-Module:

Variante 1	Single Power Supply am XM1. XF2/Pin 1 und 3 wird intern von XM1 versorgt. +24 V gelangen über eine interne 1 A Sicherung an XF2/Pin 1. XM1/A2 ist verbunden mit XF2/Pin 3.
Variante 2	Power Supply am XM1 und Logic Supply am XF2/Pin 1 und 3. Bei der Anschaltung von Spannung und Signalen an XF2 (siehe Abbildung unten) achten Sie bitte auf die Implementierung der Schutzdiode (wie in der folgenden Zeichnung verdeutlicht).

Tabelle 4

Schutzdiode XF2





Bezeichnung der Buchseneingänge der PowerCube™-Module ab Versionsnummer 3.5.00.

3.7. Elektrische Schnittstellen des Single Axis Controller PSAC

Der PSAC ist ein Controller zur Ansteuerung von 3-Phasen Wechselstrommotoren bis zu einer Leistungsaufnahme von 125 W.
Für eine hohe Störfestigkeit auch über längere Distanzen erfolgt die Anschaltung von Kontroll- und Feedbacksignalen differenziell über geschirmte twisted pair- Leitungen.
Zu den Feedback-Signalen zählen:

- die Hallgeber-Signale zur Steuerung der Kommutierung
- die Signale vom Inkrementalgeber zur Lage- und Geschwindigkeitsreglung.

Für Kontrollsignale können z. B. ein Referenz- und zwei weitere Endlagenschalter eingesetzt werden.
Die Übertragung von Bewegungs-, Steuer- und Statusbefehlen erfolgt über ein CAN-Interface.
Eine Umprogrammierung der Konfiguration bzw. des Betriebssystems ist sowohl über das CAN-Interface als auch direkt über eine ebenfalls vorhandene RS232-Schnittstelle möglich.
Die drei Leistungskabel zum Motor sollten extra geschirmt sein. Der Leitungsquerschnitt ist der gewünschten Leistungsübertragung bzw. der auftretenden Stromstärke anzupassen.

Technische Daten

Bauteil	Versorgungsspannung / maximale Stromaufnahme
Steuerungsteil	+5 V/ 1,2A
	+12V/ 100mA
Leistungsteil	+24V/ 7,5 A (siehe Motordaten)

Tabelle 5

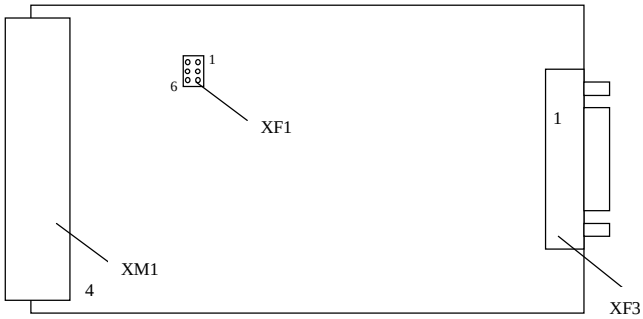
Maximaler effektiver Motordauerstrom	5A
Schaltspannung des Steuerausgangs für Bremse	+24V
Max. Dauerstrom am Steuerausgang für Bremse	500mA

Tabelle 6

Motorseitige Eingangssignale	Anzahl	Übertragung	Pegel
Hallgeber	3	differentiell	RS422
Inkrementalgeber	2	differentiell	RS422
Referenzschalter	1	differentiell	RS422
Endlagenschalter	2	differentiell	RS422

Tabelle 7

Ansicht (vereinfacht)



Pinning XM1:

Pin	Bezeichnung
4	+24 V Leistung
8	0 V Leistung
10	+12 V Leistung
12	+5 V Leistung
14	GND Steuerung
16	ERDE (Filterglied der Leistungsversorgung)
18	CAN Hi
20	CAN Lo
22	Schirm (CAN)
6,24,26,28,30,32	nicht beschaltet

Tabelle 8

Pinning XF1 (RS232)

Pin	Bezeichnung
1, 6	TxD
2, 5	RxD
3	Umschaltkontakt zur Programmierung im Bootstrapmodus
4	GND

Tabelle 9

Pinning XF3

Pin	Bezeichnung
1, 14	Phase U
2, 15	Phase V
3, 16	Phase W
4	Bremsschaltleitung (open Collector oder 0V vom Leistungskreis)
17	+24 V (vom Leistungskreis XM1/4)

Tabelle 10

5	0V (vom Leistungskreis XM1/8)
18	Hallsignal /C
6	Hallsignal C (Wittenstein braun)
19	Hallsignal /B
7	Hallsignal B (Wittenstein orange)
20	Hallsignal /A
8	Hallsignal A (Wittenstein gelb)
21	/SWR (Referenzschalter)
9	SWR
22	/SW1 (Endlage 1)
10	SW1
23	/SW2 (Endlage 2)
11	SW2
24	/B (Inkrementsignal)
12	B
25	/A (Inkrementsignal)
13	A
Schirm	GDN Empfangsschaltung (galvanisch getrennt)

Tabelle 10

Inbetriebnahme

Der PSAC 125 kann zur Information über die Konfiguration oder deren Änderung ohne Motoranschaltung betrieben werden.
Vor dem Betrieb mit einem Antrieb sind die Daten des Konfigurationsfiles auf Richtigkeit zu prüfen und gegebenenfalls umzuprogrammieren. Dazu steht dem Anwender das Programm "Power Config" zur Verfügung.
Bei angeschlossenem Motor ist die Leistungsspannung vor den Versorgungsspannungen für den Steuerungsteil (5V und 12V) anzulegen.
Dadurch wird sichergestellt, daß die Kontroll- und Feedbacksignale auf den Differenzleitungen eingeschwungen sind und die Empfänger bei der Abfrage während der Startup-Phase keine Fehlinformationen liefern.

3.8. Konfigurationshinweis

Alle PowerCube™ -Module sind ab Werk mit Standardparametern vorkonfiguriert. Hierbei handelt es sich u. a. um folgende Einstellungen: maximale Geschwindigkeit, maximale Beschleunigung, maximaler Strom, Verfahrbereich, Getriebeuntersetzung, Moduladresse. Mit dem Programm PowerConfig können Sie bequem PowerCube™ -Module nachkonfigurieren.
Die von Ihnen geänderten Parameter werden im ROM (Read Only Memory) des Moduls gespeichert. Diese Änderungen bleiben so lange erhalten (auch nach Abschalten der Module), bis die Parameter erneut geändert werden. In der normalen Anwendung steht Ihnen diese Möglichkeit nicht zur Verfügung. Viele Parameter können auch online verändert werden. Dabei werden alle Angaben durch das Modul auf die ROMseitigen Maximalwerte begrenzt.
Die von Ihnen neu festgelegten Parameter werden in das PowerCube™ -Modul "gebrannt", d. h. die Parameter werden im ROM überschrieben. Dieses "Brennen" der Parameter ist nur mit easy*Config* möglich.

3.9. Zubehör

Elektrisch (PAE)

Master-Verbindungskabel, gerade, Länge 3m (für RS232, CAN, Profibus)
PowerCube™ Abschlußwiderstand (für RS232/CAN, Profibus und RS485)
PowerCube™ Verbindungskabel, gewendelt.
PowerCube™ Verbindungskabel, gerade.
PowerCube™ 48 V-Systemanschlußkabel, gerade, Länge 3m.
PowerCube™ Bus- und/oder Spannungsverteiler (3-fach, 4-fach, 5-fach, 6-fach)

Mechanisch (PAM)

PowerCube™ Verbindungselement, gerade.
PowerCube™ Verbindungselement, konisch.
PowerCube™ Verbindungselement, Winkel 90°.
Die nächsten beiden Zeichnungen sollen Ihnen den Aufbau der Verbindungselemente näher erläutern.

Steuerung

Die Schnittstellenkarte oder Schnittstellenhardware in Ihrem Computer wird im allgemeinen als Master bezeichnet.
PCM PowerCube™ CAN-Master: Kommunikationsinterface für den PC, einschließlich Treibersoftware für Win95 / NT.
PPM PowerCube™ Profibus DP-Master: Kommunikationsinterface für den PC, einschließlich Treibersoftware für Win95 / NT.

Digitale I/O-Module

Die CAN-Knoten der CST-Baureihe sind für denzentrale Ein-/Ausgabe in Meß-, Steuerungs- und Automatisierungsanwendungen vorgesehen. Ihr kompakter und robuster Aufbau mit komplett vergossener Elektronik erlaubt den Einsatz nicht nur im Schaltschrank, sondern auch in Prozeßnähe. Durch den integrierten Micro-Controller können sie Vorverarbeitungsaufgaben selbständig ausführen. Sie sind geeignet in vielen Anwendungen die Kosten für Steuerungshardware und -installation zu senken bei gleichzeitiger Steigerung der Flexibilität. CST-Knoten sind modular aufgebaut. Grundeinheit ist das Basis-Modul mit Prozessor 80C32, CAN-Controller 82C200 und 8k ROM. Die Ein-/Ausgabe-Schnittstellen werden durch ein Applikations-Modul realisiert, das auf einer Seite des Basis-Moduls in Sandwich-Bauweise angeordnet ist. Verschiedene Applikations-Module sind verfügbar. Steuerungsfunktionen lassen sich effizient durch den Einsatz einer programmierbaren Variante des Basismoduls realisieren. Weitere Informationen finden Sie in dem Dokument „Modulbaureihe CST - Anwenderhandbuch“ (csthb_d.pdf).
Merkmale

- Vielseitige Lösung für Ein-/Ausgabefunktionen am CAN-Bus
- CiA und ISO 11898 kompatible Busschnittstelle
- Automatische Baudraten-Erkennung aus einer Tabelle vordefinierter Standardwerte
- Eigenintelligenz durch Microcontroller 80C32 oder DS 80C320
- Robust durch vollständig vergossene Elektronik
- Kompakter Aufbau zum Einsatz bei beengten Platzverhältnissen
- Module für unterschiedliche Anwendungen verfügbar
- Modularer Aufbau für leichte Anpassung an Applikationsforderungen.

4. Inbetriebnahme

Bevor Sie mit der Inbetriebnahme beginnen, müssen Sie sich mit den Sicherheitshinweisen zur PowerCube™-Produktfamilie vertraut gemacht haben.

4.1. Lieferumfang

Zu jedem PowerCube™-Modul wird ein Datenträger (CD-ROM) mit der aktuellen Treiber-Bibliothek m5apiw32.dll zur Ansteuerung des Moduls sowie dieses Betriebshandbuch im pdf-Format mitgeliefert. Weiterhin gibt es für die von amtec vertriebenen CAN-Bus-Karten sowie für die RS232-Schnittstelle die Testprogramme m5apitst (Console-Programm) und PCubeDemo (fensterbasiertes Programm), um die Funktionsfähigkeit der Module zu testen. Schließlich ist im Lieferumfang auch das Konfigurationstool PowerConfig enthalten. Dieses ist in der Lage bestimmte Parameter im ROM eines Moduls zu verändern, indem diese im Konfigurationsfenster neu gesetzt und dann in das Flash-ROM des PowerCube™-Moduls "gebrannt" werden. Die oben beschriebene Software mit Dokumentation, aktuelle Updates sowie Beispielanimationen von PowerCube™-Modulen können im Internet unter der Adresse www.powercube.de heruntergeladen werden. Readme-Dateien sollten, soweit vorhanden, vor der Installation sorgfältig gelesen werden. Weitere spezialisierte Software-Bibliotheken sowie das Programm quickStep (Ablaufsteuerung im Point-To-Point-Modus) sind optional gegen Aufpreis verfügbar (siehe Zubehör). Stromversorgung und -anschlußkabel (Spezifikationen siehe Produktvarianten) sind nicht im Standardlieferumfang eines Moduls enthalten und bei Bedarf separat zu bestellen.

4.2. Kundenseitig benötigte Infrastruktur

Anforderungen an die Stromversorgung:

Gleichspannung am Ausgang	24 V ± 5 %
Gleichstrom am Ausgang	Modul-Anzahl x Modul-Nennstrom x1,2
Welligkeit der Spannung am Ausgang	Restwelligkeit < 150 mVSS Schaltspitzen < 240 mVSS
Strombegrenzung	ca. 1,2 x Nennstrom
Eingang	- Überspannungsfest - mit Einschaltstrombegrenzung
Beispielproduktreihe	SIEMENS SITOP Baureihe

Tabelle 11

Anforderungen an das Steuergerät (PC oder SPS)

Computer + Betriebssystem	Pentium-PC + Windows 9x/NT

Tabelle 12

oder

SPS mit serieller Kommunikationsschnittstelle: RS232, CAN-Bus bzw. Profi-Bus-DP	Beispiel-Produkte (PROFIBUS-DP): BOSCH CL 200 DP + BT5 Siemens SIMATIC S7-300 +OP7 Siemens SIMATIC S7-400 +OP17 (in Vorbereitung)

Tabelle 13

PowerCube™-Module können auch als Gesamtpaket mit Stromversorgung und Steuergerät erworben werden.

4.3. Aufbau, Montage und Installation

Im allgemeinen erfordern einzeln gelieferte PowerCube™-Module keine besondere Montage. Werden Montageanleitungen für Konstruktionen benötigt, die aus Modulkombinationen (Manipulator) bestehen, liefert amtec diese gesondert mit . Vor Inbetriebnahme der PowerCube™-Module muß sichergestellt werden, daß die korrekte Stromversorgung ordnungsgemäß installiert wurde, der richtige Abschlußwiderstand (sowohl an den Modulen als auch an der Schnittstellenkarte) angebracht sowie das Modul sachgemäß an einer stabilen Stützkonstruktion befestigt wurde. Zur Stromversorgung sind die Datenblätter der entsprechenden PowerCube™-Module zu konsultieren (siehe dort). Die Abschlußwiderstände werden gemäß der Schnittstellenspezifikation von amtec mitgeliefert. Bei der Befestigung müssen sowohl das Gewicht der PowerCube™-Module und der daran eventuell befestigten Nutzlast als auch die darauf wirkenden Kräfte ausreichend berücksichtigt werden. Um die Module über einen PC ansteuern zu können, muß dieser über eine entsprechende Schnittstellenkarte verfügen (siehe Schnittstellen).

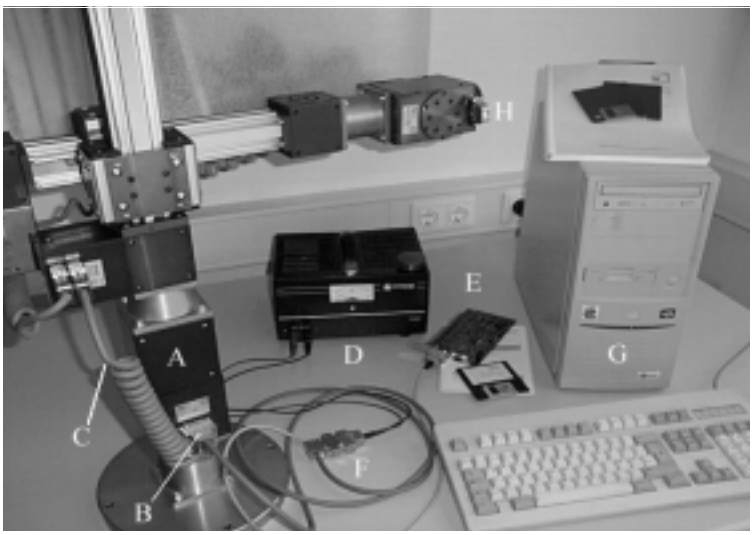


Abb. 1 zeigt exemplarisch eine Modulkombination mit der zum Betrieb notwendigen Infrastruktur (Stromversorgung, Interfacekarte, PC sowie der benötigten Verkabelung) Auf der Abbildung sind folgende Komponenten dargestellt:

- A: Modulkombination
- B: Buchse mit eingestecktem Master-Verbindungskabel (Strom- und Datenleitung)
- C: Verbindungskabel zwischen Einzelmodulen, gewendelt
- D: Stromversorgung
- E: Schnittstellenkarte (mit Installationsdiskette und -handbuch sowie Abschlußwiderstand)
- F: Master-Verbindungskabel mit eingesteckem Stromkabel (schwarz) und Datenkabel (grau; Verbindung zur Schnittstellenkarte)

- G: PC (in dem die Schnittstellenkarte eingebaut sein muß)
- H: eingesteckter Abschlußwiderstand

Die amtec GmbH bietet verschiedene Schnittstellen-Karten für unterschiedliche Bussysteme an (siehe Zubehör: Steuerungskomponenten). Es ist jedoch zu beachten, daß die Module nur für ein bestimmtes Bussystem geeignet sind und beim Wechsel auf ein anderes Kommunikationsprotokoll von amtec neu konfiguriert werden müssen.
Nachstehend wird die Installation unterschiedlicher, von amtec vertriebener CAN-Karten beschrieben.

4.4. CAN Interface iPC-I 320 (ISA-Bus)

Windows NT 4.0 Hardwareinstallation

Schritt 1:	Im Bios überprüfen, ob Interrupts (IRQs) für ISA-Erweiterungskarten reserviert wurden.
Schritt 2:	Die vom Betriebssystem verwalteten und bereits vergebenen Interrupts ermitteln (Aufruf des WinNT-Hilfsprogramms "Windows NT Diagnostics" über den Startbutton/Programs/Administrative Tools (Common)/Windows NT Diagnostics. Anwahl der Registerkarte *Resources*, Button *IRQ*).
Schritt 3:	Die vom Betriebssystem reservierten Speicherbereiche feststellen (Button *Memory*).
Schritt 4:	Die CAN-Karte auf freien Interrupt bzw. Basisadresse konfigurieren. Vom Hersteller der CAN-Karte voreingestellten Werte: - Interrupt: IRQ 5 (über Jumper einstellbar: IRQ 3, 4, 5, 7, 9, 10, 11, 12, 14, 15), - Basisadresse (reservierter Arbeitsspeicher): D000 - D1FF (über DIP-Schalter wählbar: ein Bereich von C000h - FE00h, in 8 kB-Schritten einstellbar).
Schritt 5:	Den Interrupt der CAN-Karte im BIOS für ISA-Steckkarte reservieren.
Schritt 6:	PC stromlos machen, Stecker ziehen und danach noch einmal Netzschalter betätigen (wichtig für ATX-Boards).
Schritt 7:	PC öffnen, CAN-Karte in einem geeigneten ISA-Steckplatz so befestigen, daß eine ausreichende Kontaktierung gewährleistet ist.
Schritt 8:	PC schließen, an das Stromnetz anschließen und einschalten. Sollten nach Einschalten und Booten des Systems Fehler auftreten, sind die Schritte 4 bis 7 gegebenenfalls zu wiederholen. Anderenfalls kann mit der Installation des Kartentreibers begonnen werden.

Tabelle 14

Treiberinstallation:
Falls WinNT-Treiber nicht mitgeliefert wurde (nicht auf der Diskette "iPC-I 320 - Test- und Installationsprogramm" enthalten!), ist das Programmpaket für Windows NT unter der Internet-Adresse ftp://vci:xa5i196ps@www.stzp.de/vci herunterzuladen. Es enthält neben dem erforderlichen Treiber auch nützliche Utilities wie den CAN-Minimon.

Schritt 9:	Auf der Internet-Seite das Verzeichnis *Winnt* auswählen und von dort die Datei *vcint112.exe* (vollständiges Setup-Programm) oder die beiden gepackten Archive *disk1.zip* und *disk2.zip* (um Installationsdisketten zu erstellen) in ein temporäres Verzeichnis herunterladen.
Schritt 10:	Sicherstellen, daß der Benutzer zum Zeitpunkt der Installation Administratorrechte hat.
Schritt 11:	Treiber installieren durch Ausführen der einzelnen *EXE* -Datei oder der *setup* -Routine auf Disk 1 der entpackten Installationsdisketten.
Schritt 12:	Den voreingestellte Installationspfad "c:\Programme\IXXAT\Vcint114" den eigenen Wünschen anpassen.
Schritt 13:	Den Computer neu starten.
Schritt 14:	Zum Test der Funktionsfähigkeit des Treibers das neu installierte Programm *Minmon32* starten.

Tabelle 15

Schritt 15:	Im Dialogfenster *Board Type* die eingebaute CAN-Karte sowie ihre Einstellungen (Segment: Basisadresse, IRQ Nr.: Interrupt) auswählen und den OK-Button drücken. Im Protokollfenster (*VCI-Status*) sollten alle Tests mit OK quittiert werden, danach schließt sich dieses Fenster selbstständig. Alle Anzeigen in der Status-Box im Fenster *CAN-MINIMON1* sollten auf grün wechseln.
Schritt 16:	Beim Auftreten von Fehlern werden im Hauptmenü der Menüpunkt *Config/Boardtype* ausgewählt und alle Einstellungen im *Board Type*-Fenster überprüft.
Schritt 17:	Liegen dort keine Fehleingaben vor, so sind gegebenenfalls andere Interrupts bzw. Basisadressen auf ihre Verwendbarkeit zu testen. Zum Auffinden einer freien Basisadresse kann das Hilfsprogramm "TCFRTEND" verwendet werden

Tabelle 15

Win95/98 Hardwareinstallation

Schritt 1:	Im Bios überprüfen, ob Interrupts (IRQs) für ISA-Erweiterungskarten reserviert wurden.
Schritt 2:	Win 95: Systemresourcen (Interrupts, belegte Speicherbereiche) im *Device Manager* (Kontextmenü von *Computer*), Registerkarte *Resources*, anzeigen lassen. Win 98: Systemresourcen über *Settings/Control Panel/System*, Registerkarte *Device Manager* einsehen. Dann freien Interrupt wählen.
Schritt 3:	Die ausgewählte Basisadresse in der system.ini (im Ordner *Windows*) unter dem Unterpunkt [386ENH] eintragen (z. B. mit Notepad), beispielsweise: EMMExclude = D000-D1FF
Schritte 4 bis 8:	Siehe "Installation unter Windows NT 4.0"

Tabelle 16

Treiberinstallation:

Schritt 9:	Falls nicht beiliegend, kann die Treibersoftware unter der Internet-Adresse ftp://vci:xa5i196ps@www.stzp.de/vci aus dem Ordner *Win95* auf ein beliebiges Verzeichnis des eigenen Rechners heruntergeladen werden.
Schritt 10:	Treiber installieren durch Ausführen der einzelnen *EXE*-Datei oder der *setup*-Routine auf Disk 1 der entpackten Installationsdisketten.
Schritt 11:	Den voreingestellte Installationspfad "c:\Programme\IXXAT\Vcint11x" den eigenen Wünschen anpassen.
Schritt 12:	Den Computer neu starten.
Schritt 13:	Zum Test der Funktionsfähigkeit des Treibers das neu installierte Programm *Minmon32* starten.
Schritt 14:	Im Dialogfenster *Board Type* die eingebaute CAN-Karte sowie ihre Einstellungen (Segment: Basisadresse, IRQ Nr.: Interrupt) auswählen und den OK-Button drücken. Im Protokollfenster (*VCI-Status*) sollten alle Tests mit OK quittiert werden, danach schließt sich dieses Fenster selbstständig. Alle Anzeigen in der Status-Box im Fenster *CAN-MINIMON1* sollten auf grün wechseln.
Schritt 15:	Beim Auftreten von Fehlern werden im Hauptmenü der Menüpunkt *Config/Boardtype* ausgewählt und alle Einstellungen im *Board Type*-Fenster überprüft.
Schritt 16:	Liegen dort keine Fehleingaben vor, so sind gegebenenfalls andere Interrupts bzw. Basisadressen auf ihre Verwendbarkeit zu testen. Zum Auffinden einer freien Basisadresse kann das Hilfsprogramm "TCFRTEND" verwendet werden

Tabelle 17

4.5. CAN Interface iPC-I 165 PCI und iPC-I 320 PCI (PCI-Bus)

Installation

Die Installation der PCI-Interfaces läuft unter Windows 9x/NT 4.0 prinzipiell gleich ab.

Schritt 1:	PC stromlos machen, Stecker ziehen und danach noch einmal Netzschalter betätigen (wichtig für ATX-Boards).
Schritt 2:	PC öffnen, CAN-Karte in einem geeigneten PCI-Steckplatz so befestigen, daß eine ausreichende Kontaktierung gewährleistet ist.
Schritt 3:	PC schließen, an das Stromnetz anschließen und einschalten. Sollten nach Einschalten und Booten des Systems Fehler auftreten, sind die Schritte 1 bis 3 gegebenenfalls zu wiederholen. Anderenfalls kann mit der Installation des Kartentreibers begonnen werden.

Tabelle 18

Treiberdeinstallation

Sollte sich auf Ihrem Rechner bereits ein installierter Treiber (z. B. VCI-Version 112) befinden, so sollte dieser zuerst entfernt werden. Bevor Sie dies tun, müssen Sie aber unbedingt die bereits installieten „Devices“ auf disable schalten, da sonst die Treiber nicht sauber deinstalliert werden! Aufruf über den Startbutton/Settings/Control Panel/ Devices (Doppelklick). Dort müssen Sie für jedes Xat...-Device den Button HW Profiles klicken und in dem erscheinenden Fenster den Button Diasable klicken.
Anschließend können Sie über Startbutton/Settings/Control Panel/Add-Remove Software (Doppelklick) die bereits installierte Software entfernen.

Treiberinstallation

Schritt 4:	Falls Win9x/NT-Treiber nicht mitgeliefert wurde, Programmpacket für Windows 9x bzw. Windows NT unter der Internet-Adresse ftp://vci:xa5i196ps@www.stzp.de/vci herunterladen. Es enthält neben dem erforderlichen Treiber (VCI 114) auch nützliche Utilities wie den CAN-Minimon.
Schritt 5:	Treiber installieren durch Ausführen der einzelnen *EXE* -Datei oder der *setup* -Routine auf Disk 1 der entpackten Installationsdisketten.
Schritt 6:	Den voreingestellte Installationspfad "c:\Programme\IXXAT\Vcixx114" den eigenen Wünschen anpassen.
Schritt 7:	Den Computer neu starten.
Schritt 8:	Zum Test der Funktionsfähigkeit des Treibers das neu installierte Programm *Minmon32* starten.
Schritt 9:	Im Dialogfenster *Board Type* die eingebaute CAN-Karte sowie ihre Einstellungen (Board Nr. : Zähler des eingebauten VCI-Boards; bei nur einem eingebauten Board immer 0) auswählen und den OK-Button drücken. Im Protokollfenster (*VCI-Status*) sollten alle Tests mit OK quittiert werden, danach schließt sich dieses Fenster selbstständig. Alle Anzeigen in der Status-Box im Fenster *CAN-MINIMON1* sollten auf grün wechseln.
Schritt 10:	Beim Auftreten von Fehlern wird im Hauptmenü der Menüpunkt *Config/Boardtype* ausgewählt und alle Einstellungen im *Board Type* -Fenster werden überprüft. Beim Austausch einer älteren ISA-Karte muß weiterhin sichergestellt sein, daß sich keine Überreste des alten VCI 112-Treibers im Systemverzeichnis befinden (siehe Treiberdeinstallation). Machmal hilft es wenn Sie in diesem Fall in das Installationsverzeichnis wechseln. Dort gibt es einen Pfad namens „Sysfiles“. In diesem Pfad befinden sich einige *.reg Dateien. Klicken Sie diese doppelt, so werden die Treiber in der Registry eingetragen (falls dies bei der Installation fehlgeschlagen ist). Anschließend müssen Sie Ihren Rechner neu booten.

Tabelle 19

ACHTUNG: Der Initstring für den Zugriff auf PowerCube™-Module über PCI-Interfaces ändert sich: VCI:x,0,1 (VCI: ID der PCI-Karte [iPC-I 165 PCI = 6, iPC-I 320 PCI = 7], Zähler der eingebauten Boards, 1=> Interrupt wird automatisch ausgewählt).

4.6. CAN-Interface TinCAN (PCMCIA)

Sollen PowerCube™-Module auch an einem Laptop über eine CAN-Schnittstellenkarte angesteuert werden, so bietet sich die Verwendung der PCMCIA-Karte TinCAN von IXXAT an. Die mitgelieferte Installationsdiskette enthält Treiber für DOS und Windows 95.

Windows NT 4.0

Hardwareinstallation:

Schritt 1:	TinCAN-Karte-Karte in einen freien PCMCIA-Slot stecken.
Schritt 2:	Überprüfen, ob die Karte vom PCMCIA-Controller (zu finden unter *Settings\Control Panel\PC Card(PCMCIA)*) korrekt erkannt wurde. Hier wird auch die Nummer des Slots, indem die Karte steckt, angezeigt.
Schritt 3:	Überprüfen freier Interrupts und Basisadressen wie bei Installation der iPC-I 320-ISA-Karte beschrieben.

Tabelle 20

Treiberinstallation:

Schritt 4:	Falls nicht beigelegt, Treiber aus dem Internet von der Adresse ftp://vci:xa5i196ps@www.stzp.de/vci, Ordner WinNT, herunterladen und installieren.
Schritt 5:	Im CAN-Minimon unter Board Type das Board tinCAN und den korrekten PCMCIA-Slot einstellen.
Schritt 6:	Test durchführen, wie bei der Installation der iPC-I 320-ISA-Karte beschrieben.

Tabelle 21

Windows 95/98

Hardwareinstallation:

Schritt 1:	TinCAN-Karte in einen freien PCMCIA-Slot stecken.
Schritt 2:	Im Bios überprüfen, ob Interrupts (IRQs) für ISA-Erweiterungskarten reserviert wurden.
Schritt 3:	Win 95: Systemresourcen (Interrupts, belegte Speicherbereiche) im *Device Manager* (Kontextmenü von *Computer*), Registerkarte *Resources*, anzeigen lassen. Win 98: Systemresourcen über *Settings/Control Panel/System*, Registerkarte *Device Manager* einsehen. Dann freien Interrupt wählen.
Schritt 4:	Die ausgewählte Basisadresse in der system.ini (im Ordner *Windows*) unter dem Unterpunkt [386Enh] eintragen (z. B. mit Notepad), beispielsweise: EMMExclude = D000-D1FF

Tabelle 22

Treiberinstallation:

Schritt 5:	Den Inhalt der Installationsdisk in ein beliebiges Verzeichnis der Festplatte kopieren (Speicherbedarf ca. 2MB!):
Schritt 6:	Die Datei u*npack.bat* in einer DOS-Box starten (generiert mehrere Unterverzeichnisse mit Dateien).
Schritt 7:	Im Verzeichnis *Driver\Win95* über das Kontextmenü (rechter Mausklick, Untermenüpunkt *installieren*) der Datei *TC2_NEW.INF* die Installation des Kartentreibers durchführen.
Schritt 8:	Im Verzeichnis *Test* in einer DOS-Box die Datei *Test.bat* ausführen.
Schritt 9:	Beim Fehlschlagen dieser Testroutine das Programm *PCM_VIEW.EXE* ausführen, das ausführlichere Informationen über den Fehlerstatus der Karte bzw. des Softwaretreibers liefert.

Tabelle 23

Schritt 10:	Zur Behebung eventuell auftretender Fehler unterschiedliche Interrupts und insbesondere mehrere verschiedene Basisadressen testen:
Schritt 11:	Wie bei der Installation unter Windows NT beschrieben, kann auch das oben beschriebene VCI-Programmpacket mit CAN-Monitor dazu verwendet werden, einen gültigen Treiber zu installieren und die Karte auf ihre Funktionsfähigkeit zu überprüfen.

Tabelle 23

Wichtig: Die PowerCube™-DriveLib m5dll unterstützt nur die VCI-Software bis zur Version 112, d. h. daß die über die Vci114.exe-Dateien installierten Treiber mit der aktuellen m5dll -Bibliothek Fehlerzustände hervorrufen. Ab VCI-Treiberversion 114 wird eine spezielle Bibliothek VCI_W32.dll benötigt, um die CAN-Befehle korrekt umzusetzen. Diese Datei ist nicht mit der vom VCI-Treiber im Systemverzeichnis installierten gleichnamigen Dll identisch. Sie wird mit allen amtec-Softwareprodukten mitgeliefert und muß im gleichen Verzeichnis wie die amtec-Anwendung liegen.

4.7. RS232-Schnittstelle

Um die PowerCube™-Module über eine RS232-Schnittstelle betreiben zu können, ist lediglich eine freie serielle Schnittstelle am Steuer-PC erforderlich.

Wichtig

Es ist darauf zu achten, daß es zwei Typen von Abschlußwiderständen gibt! Typ I unterstützt CAN/Profibus/RS232-Interfaces, Typ II unterstützt RS 232-Schnittstellen.
Zur Ansteuerung der PowerCube™-Module wird eine Funktionsbibliothek in Form einer *dll* mitgeliefert. Diese wird benötigt, um das Testprogramm *m5apitst* auszuführen und gegebenenfalls eigene Steuerungsprogramme für die Module entwickeln zu können. Die Funktionsbibliothek (m5dll, Stand: 21.05.99) wird installiert, indem die Datei setupex.exe im Verzeichnis m5apiw32 ausgeführt wird und den Anweisungen auf dem Bildschirm gefolgt wird. Standardmäßig wird die *dll* in das Verzeichnis c:\Programme\amtec GmbH\M5DLL (englische Installation: c:\Program Files\amtec GmbH\M5DLL) kopiert. Werden kundenseitig eigene Programme zur Steuerung der PowerCube™-Module auf Basis der Funktionsbibliothek realisiert, so ist die Datei m5apiw32.dll in das Installationsverzeichnis dieser Software zu kopieren, um deren Funktionsfähigkeit zu garantieren. Bei der Installation der Funktionsbibliothek wird gleichzeitig auch das Testprogramm *m5apitst* mitinstalliert.
Das Konfigurationstool PowerConfig wird installiert, indem die Datei setup.exe im Verzeichnis PowerConfig ausgeführt wird und den Anweisungen auf dem Bildschirm gefolgt wird.

4.8. Die CAN-Kartenkennung

IXXAT-Karten

Die Zusammenarbeit der Schnittstelle mit der *m5apiw32.dll* erfordert die Angabe eines Initstrings, der unter anderem bei VCI-CAN-Interfaces der Firma IXXAT die Kartenkennung beinhaltet (siehe "Der Aufbau des Basis-Initialisierungsstrings").
Typische Initstrings sind z. B.:

ISA-Interface-Karte:	VCI:1,d000,5
PCI-Interface-Karte:	VCI:6,0,1

Darin bedeuten:

VCI	Schnittstelle wird als VCI-CAN-Interface der Firma IXXAT identifiziert
1	Kartenkennung (bei ISA-Interface-Karten (Kartenkennung 1 und 2):
d000	ISA-Karten: Basisadresse der CAN-Karte PCI-Interface-Karten: Boardnummer (wird nur eine Interface-Karte benutzt: ist die Boardnummer immer 0.)

5	Interrupt, den die Karte belegt (nur bei ISA-Boards; bei PCI-Boards grundsätzlich auf 1 oder gar nicht gesetzt)

Tabelle 24

VCI (IXXAT)

0	iPC-I 165
1	iPC-I 320
2	CANdy 320
3	tinCAN V2
4	PCI 01, PCI 02, PCI 03
5	iPC-I 386iPC-I 386iPC-I 386
6	iPC-I 165 PCI
7	iPC-I 320 PCI

Tabelle 25

Für die passive Karte i-PC03 wird unter Linux folgender Initialisierungsstring benötigt: „CAN4LINUX:X,0“ (X = Busnummer).

CAN-AC2-PCI (Vector) Die zur Zeit unterstützte PCI-CAN-Karte der Firma Vector (CAN-AC2-PCI) benötigt keinen Identifier und kann mit folgenden Initialisierungsstrings angesprochen werden:
Windows (Win NT/9x): „VECTOR:0,0“
Linux: „VECTOR:X,0“ (X = Busnummer)

C331 PCI (ESD) Die PCI-CAN-Karte C331 von ESD benötigt folgende Initialisierungsstrings:
Windows (Win NT/9x): „ESD4WIN:0,0“
QNX: „ESD4QNX:X,0“ (X = Busnummer)

OR Computer Unter Linux wird folgender Initialisierungsstring benötigt: „ORCAN:0,0“.

4.9. Normalbetrieb

Vor Inbetriebnahme der Module, insbesondere von Modulkombinationen (Manipulator), ist ihr Aktionsradius zu ermitteln und zu kennzeichnen. Es ist dafür zu sorgen, daß sich während des Betriebs der PowerCube™-Module niemand im gekennzeichneten Gefahrenbereich aufhält.
Die PowerCube™-Module müssen in regelmäßigen Abständen gewartet und bestimmte Modulteile auf betriebsbedingte Veränderungen untersucht werden (siehe dazu im Kapitel Wartungshinweise).
Weitere Hinweise finden sich im Kapitel Betriebssystem: Betriebsweise im Normalfall.

4.10. Besondere Betriebssituationen

Bei Modulen ohne Bremse ist darauf zu achten, daß nach Abschalten des Antriebs die Lage der Module durch eventuell daran befestigte Lasten verändert werden kann.

4.11. Betriebsstörungen und deren Beseitigung

Das Betriebssystem der PowerCube™-Module verfügt über eine interne Verwaltung des Antriebsstatus. Dieser Status kann mit dem Statuswort CubeState abgefragt werden. Die Bedeutung der einzelnen Bits im Statuswort ist im Kapitel

Kommunikationsschnittstelle, Rubrik "Betriebsweise im Normalfall" nachzuschlagen. Darin werden auch mögliche Strategien zur Fehlerbehebung vorgeschlagen. Es ist ratsam, dieses Statuswort für die einzelnen Module regelmäßig zu abzufragen (polling), um auf eventuell auftretende Fehlerzustände reagieren zu können.
Beim Auftreten von Fehlerzuständen, die nicht über das Statuswort diagnostiziert werden können, empfiehlt es sich, das Modul abzuschalten (stromlos zu machen). Zunächst sollte nach erkennbaren physischen Fehlerquellen (Hindernis im Fahrtweg, Stromführung beschädigt u. ä.) gesucht werdenen. Führen diese nicht zur Beseitigung des Fehlerzustands, ist es ratsam, sich zur Behebung an amtec oder einen seiner Servicepartner zu wenden.
Weitere Hinweise siehe im Kapitel Betriebssystem unter der Rubrik "Betriebsweise bei Störungen".

4.12. Ausserbetriebnahme

Bei der Außerbetriebnahme sind die Hinweise für besondere Betriebssituationen zu berücksichtigen.

5. Betriebshinweise

Die Beachtung der nachfolgenden Hinweise ist für die Lebensdauer der Produkte entscheidend.

5.1. Hinweise zur Pflege der PowerCube™-Produkte

Die PowerCube™-Produktfamilie ist für den industriellen Einsatz entwickelt worden und aus diesem Grund sehr robust.
Reinigen Sie den PowerCube™ in regelmäßigen Abständen mit einem trockenen Tuch. Entfernen Sie grobe Verschmutzungen, die sich möglicherweise in Hohlräumen und an Kanten bilden. Achten Sie darauf, daß offenliegende Zahnriemen nicht mit Schmutz zugesetzt werden, weil dies die Funktion und Positioniergenauigkeit des PowerCube™ beeinträchtigen kann.

5.2. Oberfläche der PowerCube™-Antriebsmodule

Die Gehäuse der PowerCube™-Antriebsmodule werden aus Aluminium gefertigt. Sie haben eine hohe Oberflächenqualität, die durch eine Oxidschicht veredelt wird. Die Oberflächengüte der PowerCube™-Antriebe ist für ihre Verwendung in Manipulatoren und Robotersystemen von hoher Bedeutung, weil die geometrische Ausrichtung von der Oberflächengüte abhängig ist.
Die Veredelung durch die PowerCube™-blaue Oxidschicht ist somit ein wesentliches Funktionsmerkmal der PowerCube™-Produkte. Aus Fertigungsgründen kann es in der Farbgebung zu geringfügigen Abstufungen kommen, wodurch die Qualität des Produkts aber nicht beeinträchtigt wird.
Beachten Sie bei der Arbeit mit den PowerCube™-Antrieben, daß die Oberfläche kratzempfindlich ist. Oxidschichten mit Kratzstellen können technologisch bedingt nicht nachbehandelt werden. Im Rahmen der Fertigung werden alle PowerCube™-Module mit einer Spezialfolie weitestgehend vor Kratzwirkung geschützt. Geringfügige Kratzstellen können nicht vollständig vermieden werden und sind von der amtec-Qualitätssicherung für die Auslieferung zugelassen.

5.3. Wartungshinweise

Alle Wartungsarbeiten dürfen nur bei abgeschalteten PowerCube™-Modulen und von fachkundigem Personal durchgeführt werden, um Schäden an den Modulen zu vermeiden.

Sicherungswechsel

Derzeit sollten die Module bei einem notwendigen Sicherungswechsel zu amtec eingeschickt werden, da hierfür das Öffnen des PowerCube™-Moduls erforderlich ist. Bei neuen Modul-Baureihen wird der Wechsel über eine abnehmbare Stromzuführung realisiert. Dazu wird das Handbuch aktualisiert, das den Sicherungswechsel ausführlich beschreiben wird.

5.4. Einstellung der Endlagen- und Referenzschalter

Endlagen- und Referenzschalter können von Kundenseite nur bei Linearmodulen mit Zahnriemenantrieb sowie mit Kugelgewindetrieb neu justiert bzw. eingestellt werden.

Zahnriemenantrieb

Der Endschalternocken dient der elektrischen Endlagenbegrenzung und der

Einstellung des Referenzpunktes der Lineareinheit. Dieser Stahlstift muß einen Abstand von ca. 0,4 bis 0,8 mm vom End- bzw. Referenzschalter haben (Abb. 2). Die Module werden von amtec mit der entsprechenden Einstellung ausgeliefert. Während des Betriebs können allerdings Änderungen des Abstands auftreten, die eine Nachjustierung der Endschalter erforderlich machen.

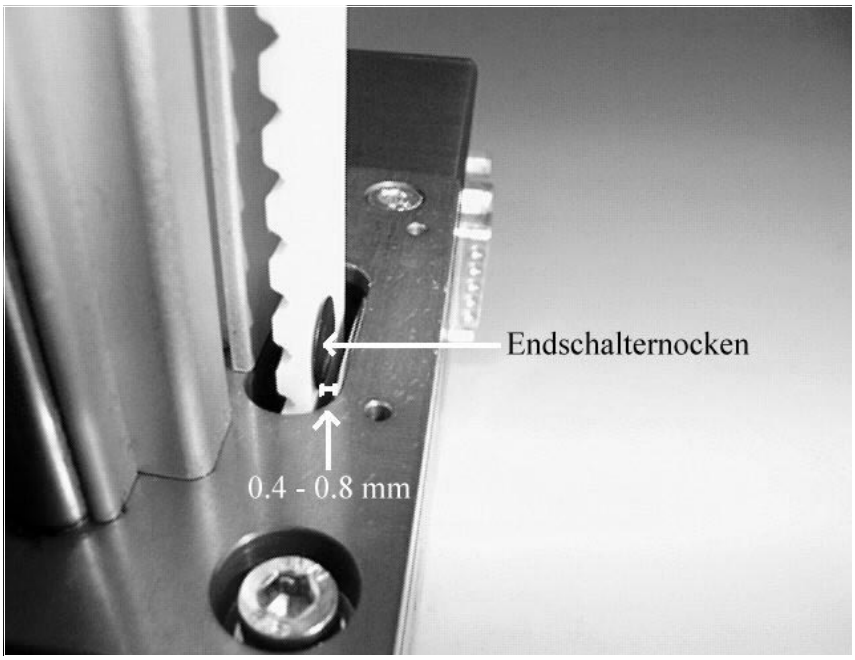


Abb. 2

Die Endlagen- und Referenzschalter befinden sich am Steuer/Antriebs-Modul C des in Abb. 3 exemplarisch dargestellten Linear-Moduls:

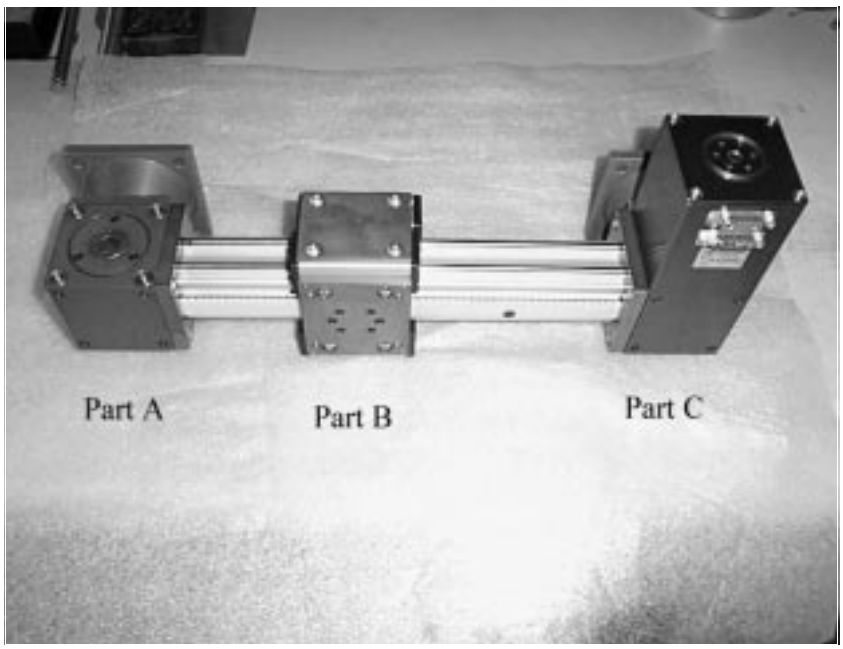


Abb. 3

Der Abstand des Endlagenschalters vom Endschalternocken kann durch Lösen der beiden Schrauben in den Positionen A und B in Abb. 4 und Drücken mit einem dünnen Stift oder Draht an Position C bzw. einem Gegendruck von der anderen Seite nachjustiert werden (Abb 4):

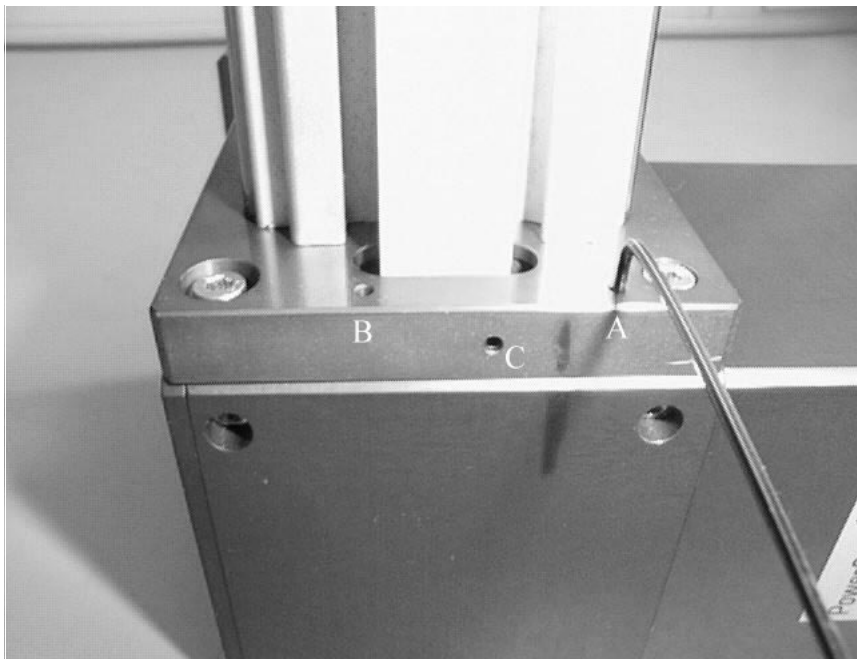


Abb. 4
Alle PowerCube™-Module werden von amtec mit eingestellten und geprüften Endlagen- und Referenz-Schaltern ausgeliefert. Das Verändern der Lagen insbesondere der Endschalternocken sollte nur von amtec vorgenommen werden und geschieht sonst auf eigene Gefahr.

Kugelgewindetrieb

Abb. 5 zeigt ein Linearmodul mit Kugelgewindetrieb in der Gesamtansicht:

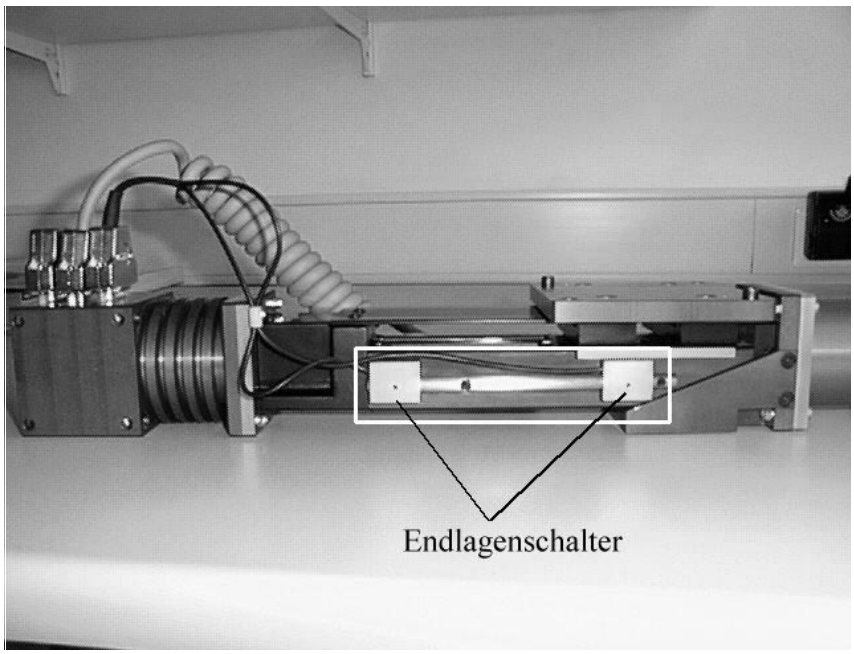


Abb. 5
Der weiß umrandete Ausschnitt zeigt die Lage der Endschalter am Modul sowie der Achse, auf der sie bewegt werden können.
Die Endlagenschalter können mit einem Imbus-Schlüssel gelöst werden (Abb. 6):

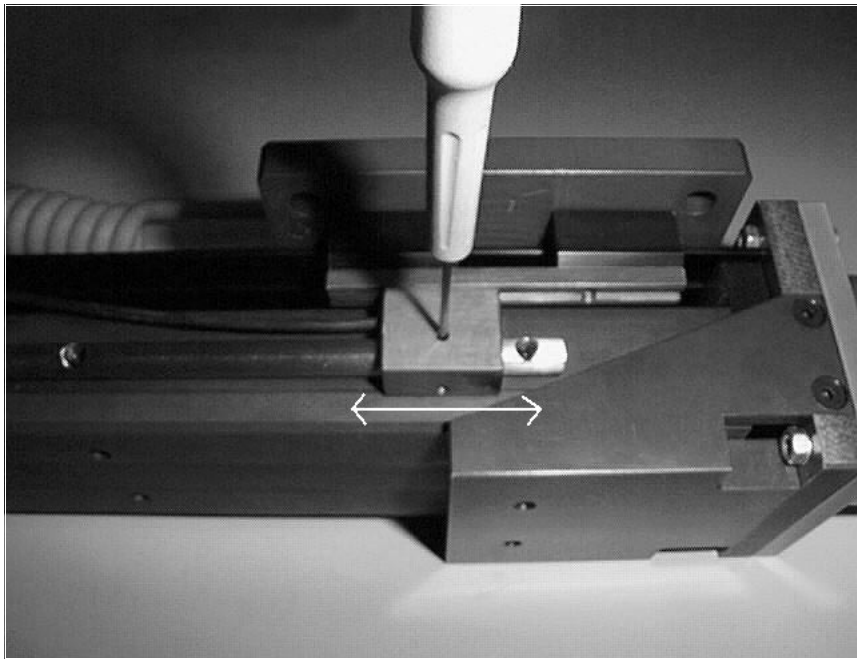


Abb. 6

Dann kann die Lage der Endschalter den Betriebserfordernissen angepasst werden. Dafür sollte unbedingt die ursprüngliche Position der Endschalter markiert werden, um gegebenenfalls die Ausgangslage wiederherstellen zu können.

Zahnriemenspannung

Modulteil A in Abb. 3 bezeichnet die Umlenkeinheit, in dem sich der Exzenter zum Spannen des Zahnriemens befindet. Zuerst müssen die 3 Schrauben gelöst werden, die den Exzenter fixieren:

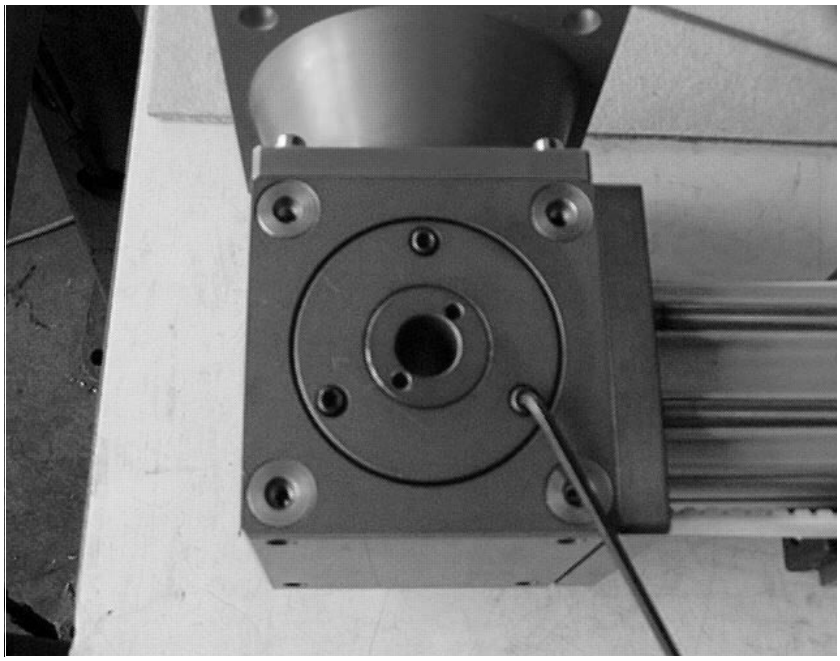


Abb. 7

Dann wird mittels einer Spezialzange oder eines anderen geeigneten Werkzeugs der Exzenter soweit gedreht (gegen oder auch im Uhrzeigersinn), bis die gewünschte Zahnriemenspannung eingestellt wurde:

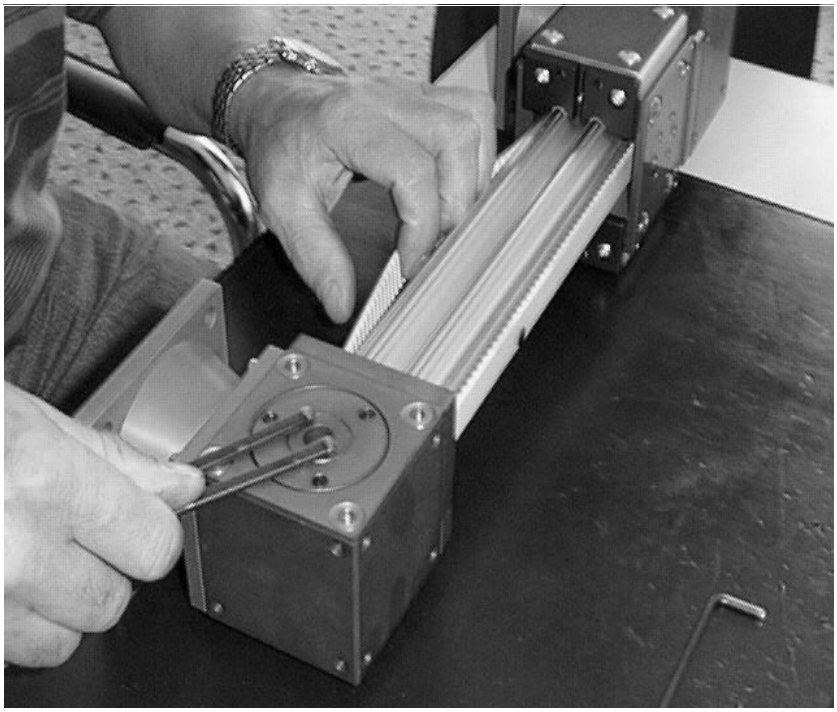


Abb. 8

Laufrollen

Es gibt die PowerCube™-Linearmodule in verschiedenen Ausführungen mit unterschiedlichen Lagereinheiten: PLB 70, PLB 90 und PLB 110(70mm, 90 mm und 110 mm Kantenlänge des Modulgehäuses). Die Linearmodule PLB 70 und PLB 90 haben pro Lagereinheit je eine Reihe fester Laufrollen (nicht verstellbar) und eine Reihe Laufrollen mit exzentrischen Bolzen, die einen Verstellbereich von ±1,5 mm ermöglichen. Diese Verstellmöglichkeit erleichtert es, eine spielfreie Führungseinheit zu realisieren und wird für die verschiedenen Modultypen im Folgenden beschrieben:

PLB 70

Am Linear Belt 70-Modul sind zunächst die Verbindungsplatten der Rollenführungen des oberen und unteren Teilschlittens zu lösen, wie in Abb. 9 dargestellt:

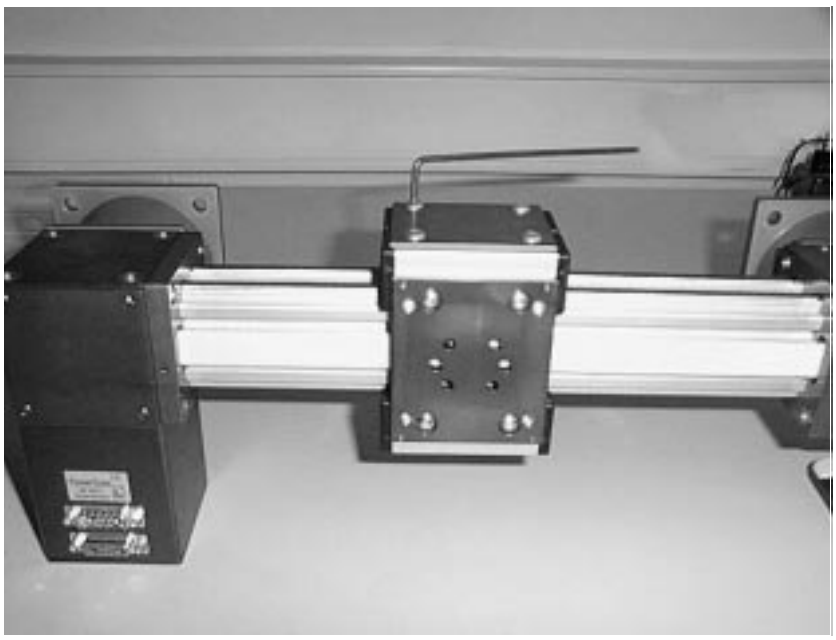


Abb. 9

Nun können die Rollenführungen auseinandergeschoben werden, wodurch die Rollenlager frei zugänglich werden (Abb. 10):

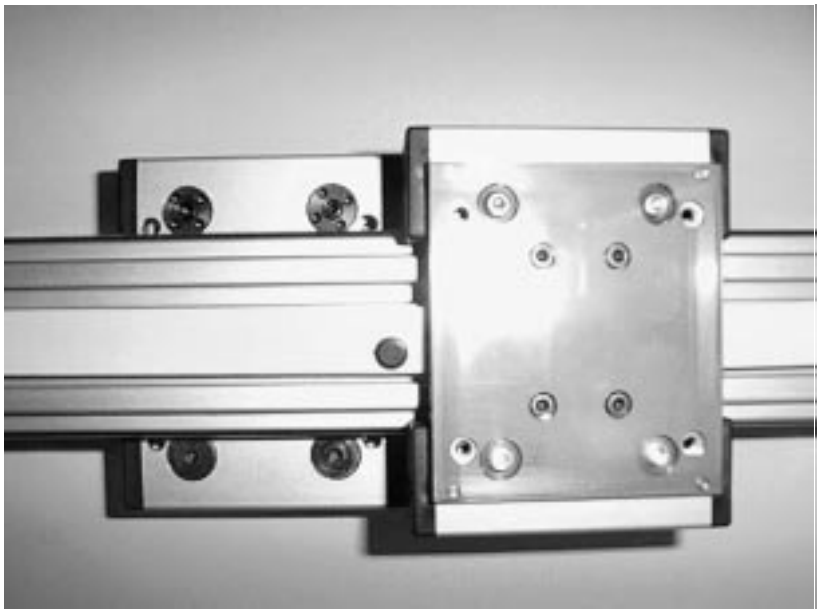


Abb. 10

Die oberen Lagereinheiten mit je einem Loch in der Mitte und vier konzentrisch angeordneten Löchern können verstellt werden. Mit einem Spezialwerkzeug (Zange mit Zapfen) wird durch Drehung die Arretierung gelöst, während mit einem Imbus-Schlüssel in der Mitte der exzentrisch angeordnete Bolzen gedreht wird. Somit kann die Lage der Rolle zur Führungsschiene verändert werden (Abb. 11).
Die Gegen-Lagereinheiten sind zentrisch und können daher nicht verstellt werden.

Abb. 11

PLB 90

Abb. 12 zeigt einen fertig montierten Schlitten eines PLB 90-PowerCube™-Linearmoduls.

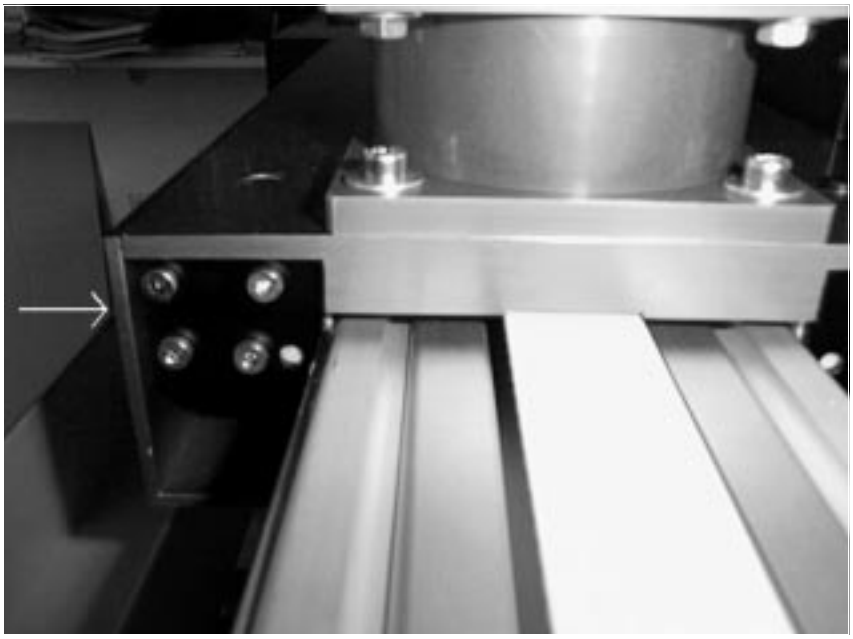


Abb. 12
Wenn die mit einem Pfeil markierte Seitenverkleidung entfernt wird, erhält man Zugang zu den dahinter befindlichen Rollen. Auch hier existieren wie beim PLB 70-Modul zwei verschiedene Lagereinheiten.
Zwei der Laufrollen sind zentrisch gelagert (z) und können nicht verstellt werden. Die auf der gegenüberliegenden Seite befindlichen Rollen dagegen lagern auf exzentrischen Bolzen (e). Abb. 13 zeigt eine Aufsicht auf die Unterseite eines demontierten Schlittens mit den entsprechend gekennzeichneten Rollen (e = exzentrisch; z = zentrisch).

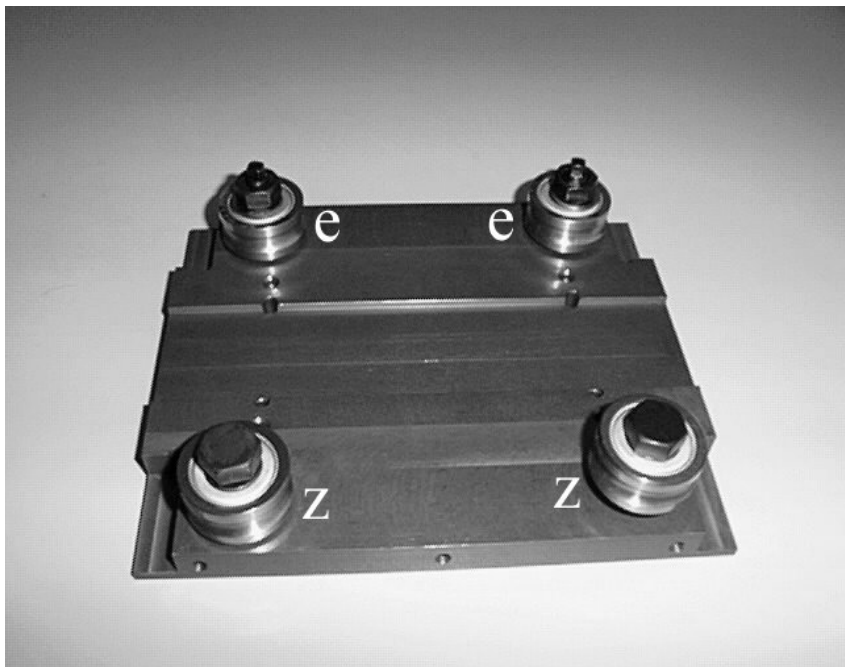


Abb. 13
Die exzentrische Rollenlagerung wird in Abb. 14 detaillierter dargestellt. Um die Lage der Rollen verstellen zu können, muß die Sechskantmutter gelöst werden; mit einem Vierkant- oder Maulschlüssel kann dann die Stellung des Exzenterbolzens variiert werden. Dazu kann der Schlitten auf der Führungsschiene verbleiben.

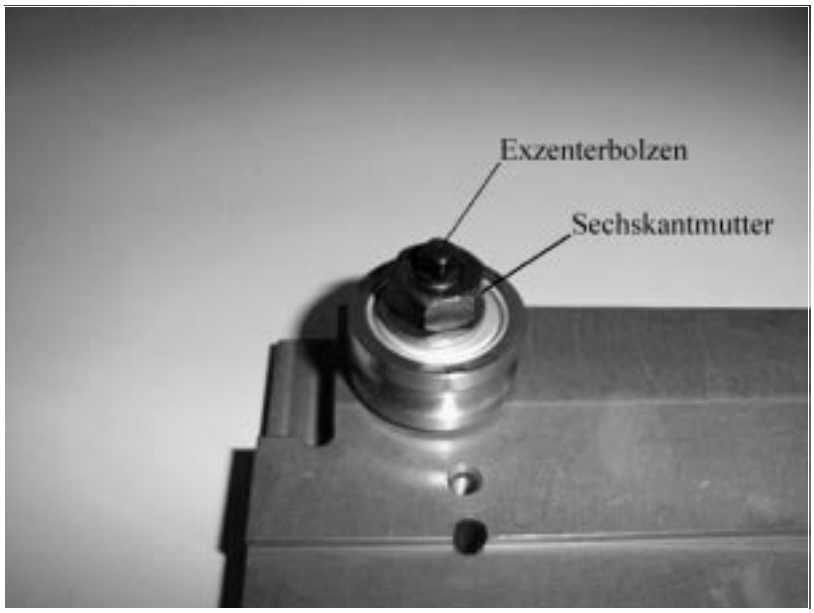


Abb. 14

5.5. Schmierung

Damit die Funktion der Linearführungssysteme nicht beeinträchtigt wird und über einen längeren Zeitraum erhalten bleibt, ist eine Schmierung entsprechend der Umgebungsbedingungen und der spezifischen Anforderungen unbedingt durchzuführen. Im allgemeinen wird eine Nachschmierfrist spätestens nach 100 km Laufleistung bzw. alle 3 Monate zugrundegelegt. Spezielle Einsatzbedingungen können diese Frist verkürzen. Es wird angeraten, bei Modulen mit Linearführungssystemen, die besonderen Belastungen unterliegen, die Führungselemente in kürzeren Abständen auf ausreichende Schmierung zu prüfen. Die Abschmierintervalle sind dann gegebenenfalls individuell den besonderen Anforderungen anzupassen. Zu berücksichtigende Faktoren sind u. a.

- Extreme Betriebstemperaturen
- Kondens- oder Spritzwassereinwirkung
- hohe Schwingungsbeanspruchung
- Einsatz im Vakuum oder Reinräumen
- Beaufschlagung mit speziellen Medien (z. B. Dämpfe, Säuren oder Kohlenwasserstoffe)
- hochdynamischer Betrieb
- permanente kleine Hubbewegungen (Hubweg < 2 Wagen-/ Mutterlängen).

Für den Einsatz unter normalen Betriebsbedingungen des Linearführungssystems werden Schmierstoffe mit folgenden Mindestanforderungen empfohlen:

Schmierstoff	DIN-Kennzeichen	DIN-Nummer	Bemerkung
Schmierfett	KP2-K	51502/51825	Lithium-Seifenfett
Schmieröl	CLP32-100	51517 Teil 3	ISO VG 32-100

Tabelle 26

Die Schmierstoffmenge ist abhängig von der Hublänge. Bei besonders langen Hüben sind kürzere Schmierintervalle oder eine größere Schmierstoffmenge erforderlich, damit der Ölfilm über die ganze Länge der Laufbahnen nicht abreißt.
Sind die Linearführungssysteme Kühlmitteln ausgesetzt, sollte ein Schmierstoff mit

einer dynamischen Viskosität von ca. 68 cst oder ein besonders emulsionsbeständiger Schmierstoff eingesetzt werden. Außerdem ist in kürzeren Intervallen und größeren Mengen nachzuschmieren. Für Linearführungssysteme, die mit hohen Belastungen und Geschwindigkeiten verfahren werden, ist eine Ölschmierung zu empfehlen.
Ein besonders emulsionsbeständiges Gleitöl (dynamische Viskosität: ca. 68 cst) ist z. B. Mobil Vectra Oil No. 2S.

5.6. Schmierung der Ölabstreifer in Linearmodulen mit Zahnriemenantrieb

PLB 70

Abb. 15 zeigt die Seitenansicht eines Schlittens. Es gibt insgesamt 8 Abstreifsysteme mit jeweils einer Schmierstelle, die in oben genannten Abständen mit Schmiermittel versorgt werden müssen.

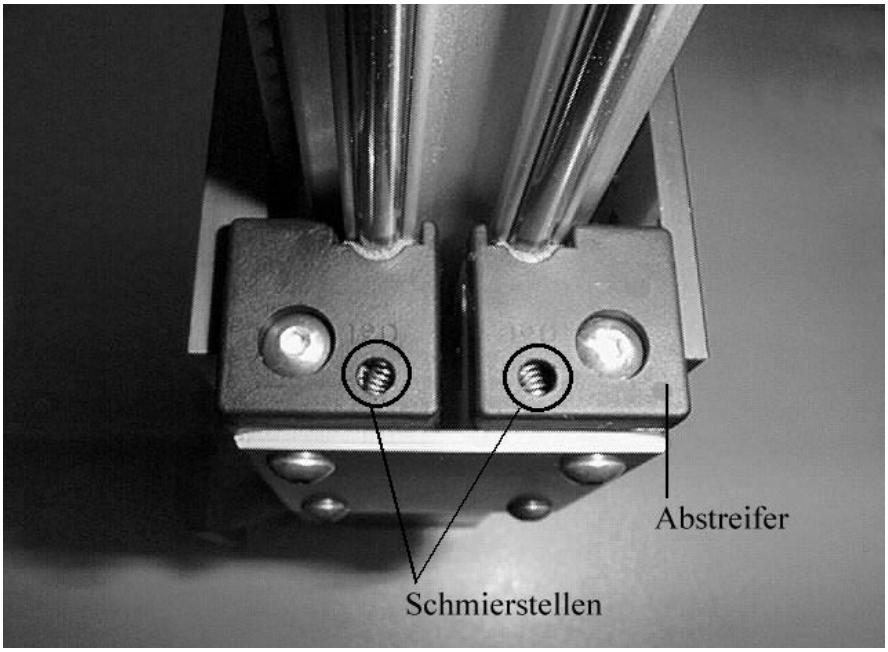


Abb. 15

PLB 90

Das Schmiermittel wird an der Stelle, die in Abb. 16 mit einem Kreis markiert ist, angewendet. Hier sind pro Schlitten vier Abstreifer mit jeweils einer Schmierstelle zu berücksichtigen.

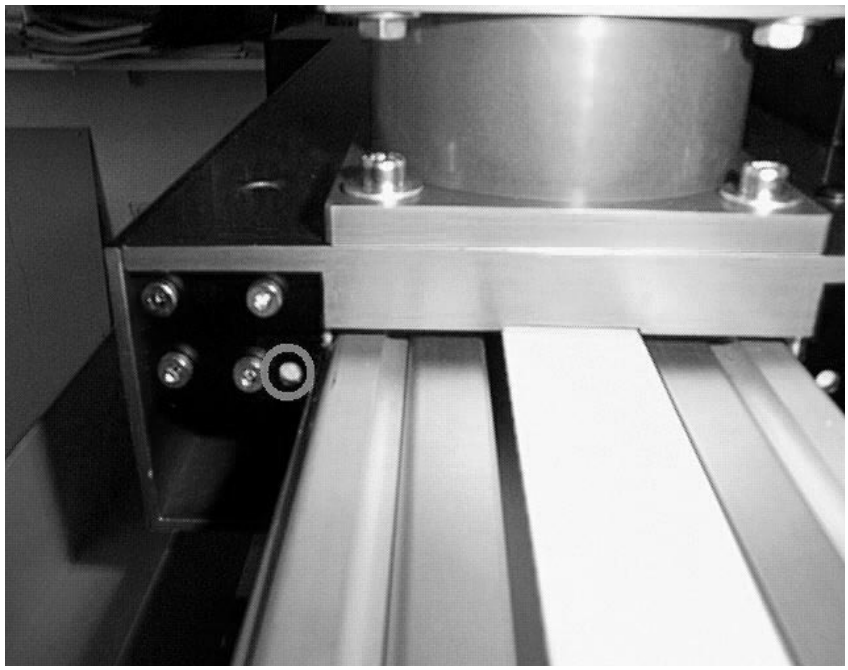


Abb. 16

5.7. Schmierung der Kugelumlaufspindel

Abb. 17 zeigt ein Linearmodul mit Kugelgewindetrieb in der Gesamtansicht:

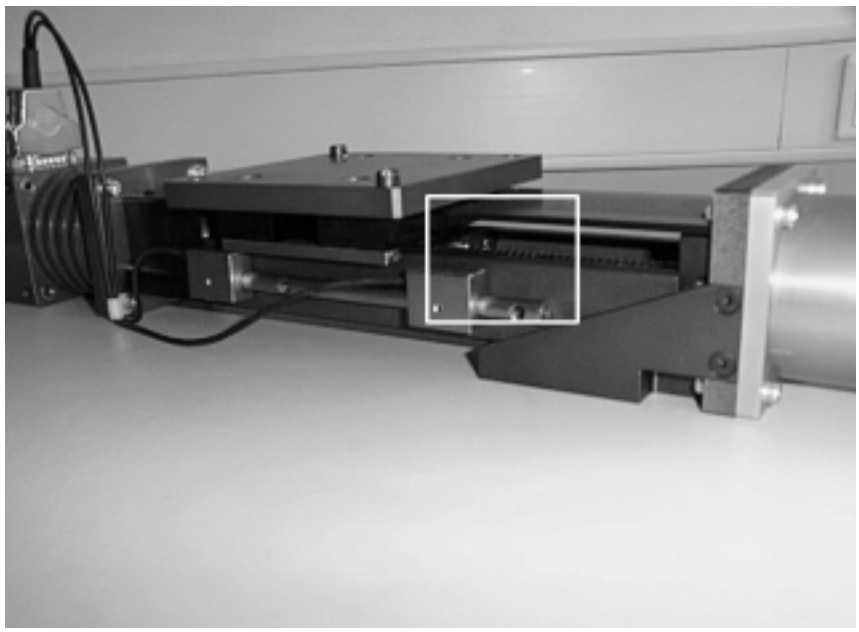


Abb. 17
Der weiß umrandete Ausschnitt in Abb. 17 enthält die Schmierstelle und wird in Abb. 18 vergrößert dargestellt:

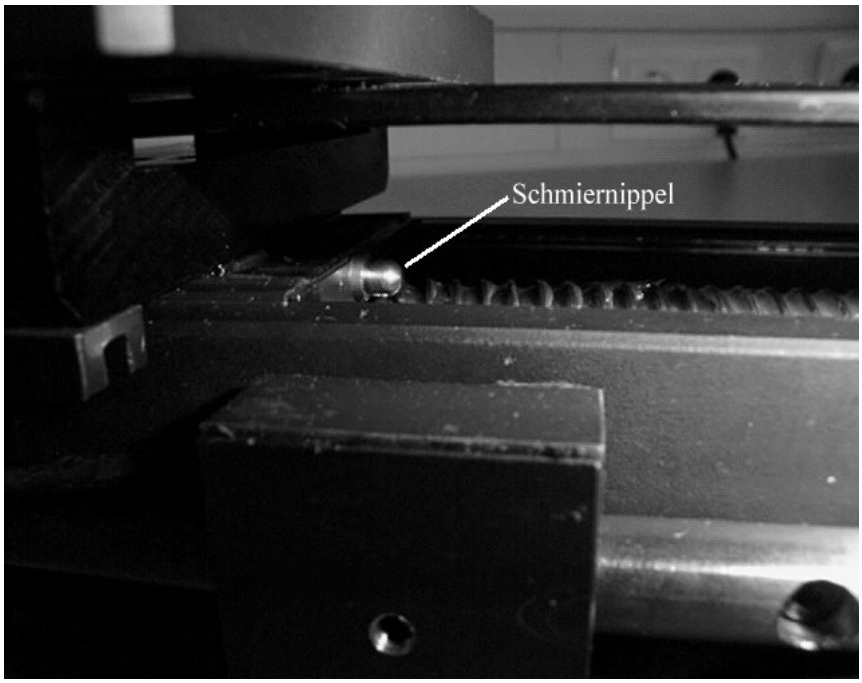


Abb. 18
Mit der Fettpresse Typ MG70 (Vertrieb: THK; Kontakt: THK GmbH, Headquarters Düsseldorf, Hubert-Wollenberg-Str. 15, D-40878 Ratingen, Tel.: ++49 +21 02 74 25-0, Fax: ++49 +21 02 74 25-2 99, E-mail: info@thk.de, Internet: www.thk.de) und den mitgelieferten Abschmieradaptern können alle Baugrößen der Linearführungssysteme in PowerCube™-Linearmodulen mit Kugelumlaufspindeln (Linear Spindel 70 / 90 / 110) abgeschmiert werden.

5.8. Schmierung der Linearführungen (am Greifer)

Abb. 19 zeigt ein 2-Finger-Parallelgreifermodul (Gripper 70 / 90) in der Gesamtansicht:

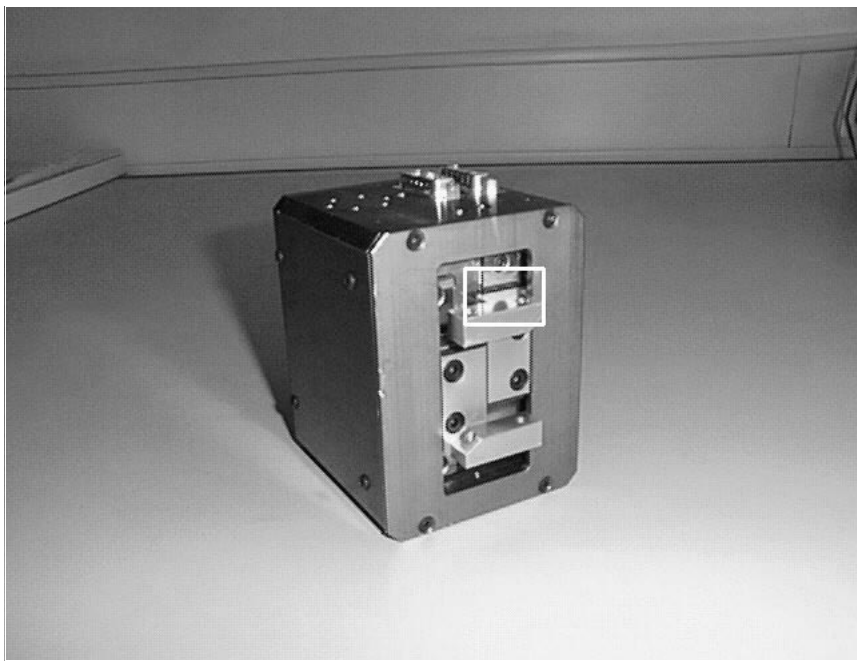


Abb. 19

Der weiß umrandete Ausschnitt in Abb. 19 enthält die Schmierstelle eines Greifers (der zweite Greifer wird an entsprechender Stelle unten abgeschmiert) und wird in Abb. 20 vergrößert dargestellt:

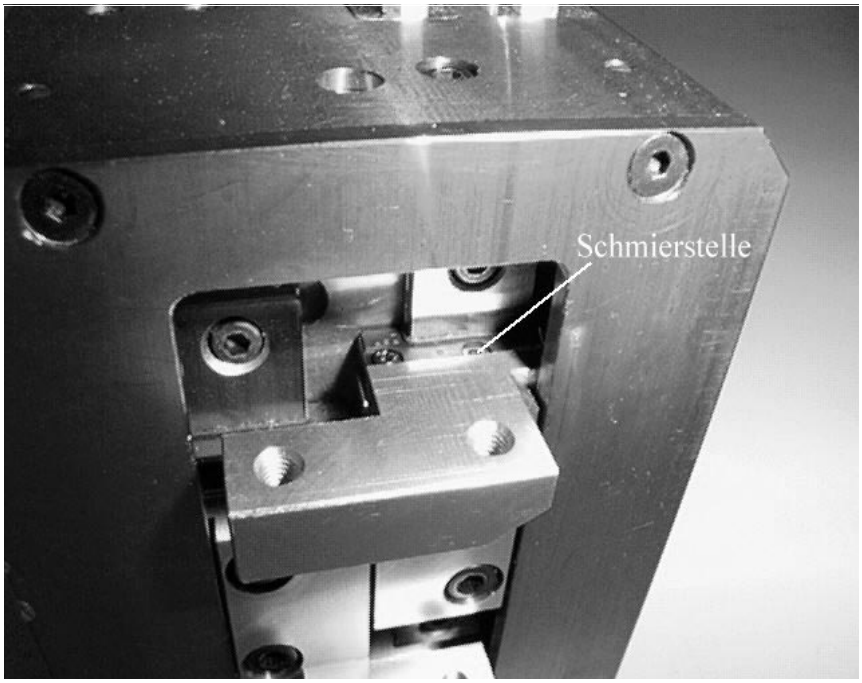


Abb. 20

An der in Abb. 20 bezeichneten "Schmierstelle" sitzt normalerweise ein Schmiernippel, ähnlich der in Abb. 18 gezeigten Schmierstelle des Linearmoduls mit Kugelgewindetrieb. Zum Abschmieren kann wieder die oben erwähnte Fettpresse Typ MG70 mit entsprechendem Adapter verwendet werden. Sind die Schmiernippel wie auf diesem Bild abgeschraubt, so genügt es auch, etwas Öl auf die Laufschienen unterhalb der Greiferbacken zu geben.

6. Konfiguration

Das Programm PowerCube™-PowerConfig ersetzt das bisher bekannte Konfigurationsprogramm PowerCube™-easy*Config* und dient der Konfiguration aller PowerCube™-Module inclusive derer mit den neuen Betriebssystemversionen ab 3.5.00.

6.1. Die Konfigurationssoftware "PowerConfig"

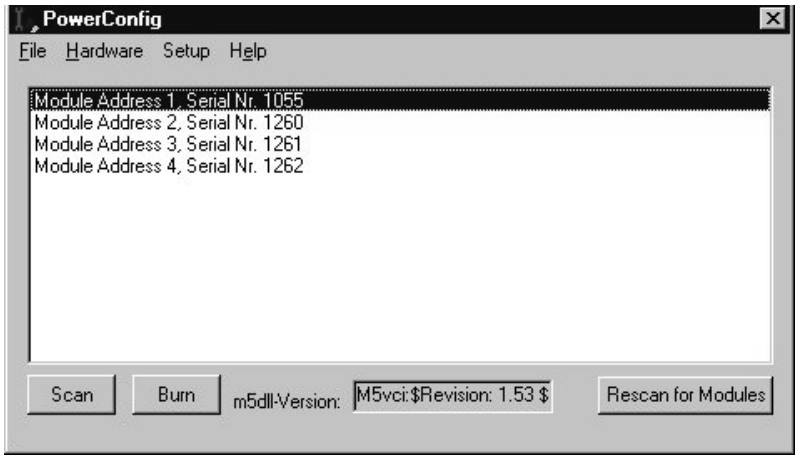
Das Produkt PowerConfig ist ein spezielles Software-Werkzeug für die PowerCube™-Produktfamilie und sollte nur von Anwendern benutzt werden, die über gute Kenntnisse der internen Parameter der PowerCube™-Module verfügen. PowerConfig unterstützt den Anwender beim Neuparametrieren von Modulen mit einer Reihe von Sicherheitshinweisen und Abfragen. Trotzdem wird dringend geraten, vor Anwendung von PowerConfig zum Brennen neuer Modulparameter Rücksprache mit amtec halten, um Schäden am Modul zu vermeiden, die durch das Brennen falscher Parameter in das Modul entstehen können. Bitte achten Sie darauf, daß die Konfiguration immer nur mit **einem** Modul am Bus durchgeführt wird! Dabei ist zu berücksichtigen, daß sich PowerCube™ Handgelenksmodule (PW) sowie Lift'n Turn-Module (PLT) wie zwei Module verhalten, da sie zwei konfigurierbare Bewegungsachsen besitzen. Weiterhin muß sowohl auf der Interface-Karte wie auf dem zu brennenden Modul ein Abschlußwiderstand angebracht sein!
Bitte beachten Sie unbedingt den Abschnitt über Fehlerbehandlung am Ende des Kapitels vor der Installation.

6.2. Installation

Stellen Sie sicher, daß vor der Installation von PowerConfig die Testsoftware für Ihre Module korrekt läuft.
Die Distribution von PowerConfig wird mit dem neuen Windows-Installer von Microsoft ausgeliefert. Sollte das Windows-Installationsprogramm noch nicht auf Ihrem Rechner installiert sein, kann es mit dem Aufruf von setup.exe installiert werden und verbleibt als Teil des Betriebssystems. Unter Umständen muß der Rechner danach neu gestartet werden.
Die eigentliche Installation von PowerConfig wird durch Doppelklicken auf die Installationsdatei PowerConfig.msi ausgelöst.
Klicken Sie 'Next' im daraufhin erscheinenden Welcome-Fenster
Im anschließenden Dialog können Sie den vorgeschlagenen Installationspfad Ihren Wünschen anpassen oder ohne Änderung übernehmen. Klicken Sie danach auf den Button 'Next'.
Danach wird überprüft, ob sich das Windows-Installationsprogramm bereits auf Ihrem Rechner befindet. Mit Betätigen des 'Next'-Buttons werden Ihre Einstellungen wirksam und die Installation beginnt.
Bitte beachten Sie die Informationen in der angezeigten Readme-Datei. Sollten Sie einen VCI-Treiber der Firma IXXAT < Version 1.14 benutzen, so löschen Sie bitte die VCI_WS32.dll aus dem Installationsverzeichnis von PowerConfig, um einen einwandfreien Betrieb des Programms zu gewährleisten.
Die Installation ist nun abgeschlossen. Klicken Sie den 'Finish'-Button.
Im Startverzeichnis von Windows wird ein Eintrag „PowerConfig“ vorgenommen. Um das Programm starten zu können, klicken Sie bitte diesen Eintrag. Sie können auch einen Shortcut auf dem Desktop erstellen (einfach das Programmicon mit Drag and Drop aus dem Fenster des geöffneten Programmverzeichnisses auf den Desktop ziehen und loslassen).

6.3. Das Hauptfenster

Das folgende Bild zeigt das Hauptfenster von PowerConfig



Das Hauptfenster von PowerConfig ist in drei Bereiche unterteilt.

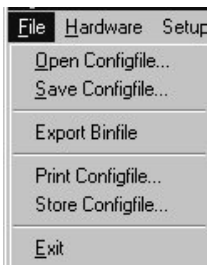
Menüleiste Der Inhalt der Menüleiste wird im nächsten Abschnitt beschrieben.

PowerCube™-Modulliste In dieser Liste finden Sie alle angeschlossenen Module. Hier finden Sie die Seriennummer und die CAN-Bus-Adresse der Module. Alle Änderungen werden für das Modul durchgeführt, das in der Liste selektiert ist (gilt nur für PW- und PLT-PowerCube™-Module). Durch Doppelklicken auf einen Listeneintrag wird der Dialog zum Ändern der Parameter für das betreffende Modul geöffnet.

Buttons Der Scan Button wird verwendet, um die angeschlossenen Module zu finden und in der Modulliste anzuzeigen. Durch erneutes Betätigen des Scan Buttons kann, unter Berücksichtigung eines eventuell zwischenzeitlich geänderten Initstrings, erneut nach Modulen gesucht werden. Der Rescan Button kann verwendet werden, um die Liste zu aktualisieren, falls Sie zur Laufzeit des Programms neue Module angeschlossen haben. Alle Änderungen die Sie bisher getätigt haben gehen verloren, falls Sie zuvor Ihre Änderung noch nicht gebrannt haben. Ein zwischenzeitlich geänderter Initstring wird nicht berücksichtigt. Mittels des Burn Buttons gelangt man in den Brenn-Dialog, in dem die geänderten Parameter dauerhaft in das Modul gebrannt werden können. In der Mitte zwischen den Buttons klärt eine Anzeige über die Version der vom Programm genutzten m5dll auf (nur, falls Scanvorgang erfolgreich war). Während des Scanvorgangs wird an dieser Stelle eine Fortschrittsanzeige eingeblendet, die den ungefähren Fortgang des Scanprozesses wiedergibt.
Bitte beachten Sie, daß beim Starten des Programms noch nicht auf die Hardware zugegriffen wird.

6.4. Die Menüleiste

Der Menüpunkt Datei Die erste Spalte in der Menüleiste ist das Dateimenü.



Mit dem Dateimenü haben Sie die Möglichkeit, Modulkonfigurationen auf einem Speichermedium zu sichern oder von dort zu laden.

Open Configfile Um eine Konfigurationsdatei zu öffnen, muß zunächst das passende Betriebssystem ausgewählt werden. Dies geschieht entweder automatisch beim erfolgreichen Scannen nach Modulen oder kann manuell durch entsprechende Auswahl im Menü Hardware (siehe dort) selektiert werden. Bei manueller Auswahl muß der Benutzer das für die Konfigurationsdatei (*.cnf) passende Betriebssystem kennen. Zwischen Konfigurationsdateien für alte Betriebssysteme (250x und 251x) und neueren Betriebssystemen (ab 350x) kann aufgrund der unterschiedlichen Dateilänge (ersichtlich z. B. im Windows Explorer) leicht unterschieden werden. Klicken Sie nun einfach auf den Menüeintrag "Open Configfile" und wählen eine Datei in der darauf folgenden Auswahlbox aus. Die geladenen Parameter überschreiben die aktuellen Parameter des dabei selektierten Moduls (nicht dauerhaft).

Beispiel Sie haben ein Modul am CAN-Bus angeschlossen und erfolgreich per Sacn erkannt. Wählen Sie anschließend im Dateimenü den Punkt "Open Configfile" und laden eine Konfigurationsdatei in den Speicher des Computers. Die Parameter des Moduls sind nun überschrieben (nur im PC). Sie können Ihre alte Konfiguration wiederherstellen, indem Sie den Button "Scan" bzw. "Rescan for Modules" drücken.

Save Configfile Dieser Menüpunkt wird anwählbar, wenn erfolgreich nach Modulen gescannt wurde oder eine Konfigurationsdatei mit dem Menüpunkt "Open Configfile" ausgewählt wurde. Wenn Sie die Konfiguration eines ausgewählten Moduls oder eine geladene Konfigurationsdatei speichern möchten (mit der Extension .cnf), wählen Sie diesen Menüpunkt. Sie können eine gespeicherte Konfigurationsdatei später auch einem anderen Modul (oder demselben Modul) zuweisen. Beachten Sie, daß die Seriennummer dabei nicht geändert werden kann. Markieren Sie einfach ein Modul und klicken dann auf "Open Configfile".

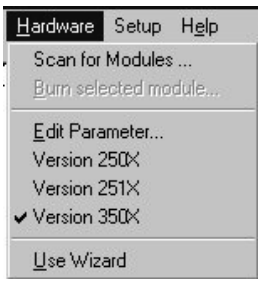
Export Binfile Dieser Menüpunkt wird anwählbar, wenn erfolgreich nach Modulen gescannt wurde. Bei dieser Option haben Sie die Möglichkeit alle Parameter und das Betriebssystem zu sichern. Diese Datei kann später verwendet werden, um Module nicht nur mit Parametern sondern auch mit dem Betriebssystem zu brennen.

Print Configfile Für Ihre Dokumentation können Sie sich die Parameter eines jeden angeschlossenen Moduls ausdrucken lassen.

Store Configfile Neu ist die Möglichkeit, die Parameter einer Konfigurationsdatei als ASCII-Datei mit der Endung .out zu speichern.

Exit Wenn Sie diesen Menüpunkt klicken, beenden Sie das PowerConfig Programm ohne weitere Meldungen.

6.5. Der Menüpunkt Hardware



Scan for Modules Der Menüpunkt wird verwendet, um die angeschlossenen Module zu finden und in der Modulliste anzuzeigen.

Burn selected module Dieser Menüpunkt ist nur zugänglich, wenn zuvor erfolgreich nach Modulen gescannt oder eine Datei geladen wurde. Wenn Sie diesen Menüpunkt auswählen, öffnet sich der Dialog "Burning Module". (siehe Abschnitt "Dialog Burning Module"). In das selektierte Modul können die Parameter (und das Betriebssystem, falls gewünscht) eingebrannt werden.

Edit Parameter Dieser Menüpunkt ist nur zugänglich, wenn zuvor erfolgreich nach Modulen gescannt wurde bzw. wenn eine Konfigurationsdatei mit "Open Configfile" geladen wurde. Er öffnet den Hauptkonfigurationsdialog, mit dem Sie einzelne Parameter des selektierten

Moduls setzen können. Alle Änderungen werden nur im Speicher des Computers gehalten und nicht im ROM des Moduls. Sie müssen Ihre Änderungen brennen, um sie als ROM-Defaults sichern zu können.
Ein Sonderfall ist die Auswahl des Menüpunkts "Use Wizard" (Häckchen gesetzt). Dies wird an entsprechender Stelle erläutert.
Der Hauptkonfigurationsdialog ist unter "Die Property Pages - Modulparameter editieren" beschrieben.

Versionserkennung

Zu jedem Modul wird Ihnen automatisch angezeigt, mit welcher Betriebssystemversion dieses Modul ausgestattet ist. Sie haben die Möglichkeit, ein Update des Modulbetriebssystems von 250X auf 251X durchzuführen. Einige wenige Parameter haben sich zwischen den Versionen 250X und 251X geändert. Ein Update der Betriebssystemversionen 25xx auf 350X ist nicht möglich und wird von PowerConfig auch nicht zugelassen. Die Auswahl des Betriebssystems durch Selektieren des betreffenden Menüpunkts bestimmt das Betriebssystem, das im Dialog "Burning Module" zum Flash-Download angezeigt wird. Folgende Auswahl steht Ihnen zur Verfügung:

- Version 250X
- Version 251X
- Version 350X (siehe Zusammensetzung der Versionsnummern)

Use Wizard

Wird dieser Menüpunkt aktiviert (Häckchen gesetzt), so kommt man beim Aufruf des Menüpunkts "Edit Parameter"in den neuen Dialog "Select Application" (siehe Benutzung des Konfigurationswizards).

6.6. Der Menüpunkt Setup

Der Menüpunkt "Setup" sieht wie folgt aus:

Config

Durch Aufruf dieses Untermenüpunkts gelangt man in den Dialog "Initialization". Dieser Dialog sollte nach Installation des Programms als erstes aufgerufen werden, da hier z. B. der Initstring und die Kommunikationsschnittstelle gesetzt werden. Einzelheiten siehe unter "Der Dialog Config".

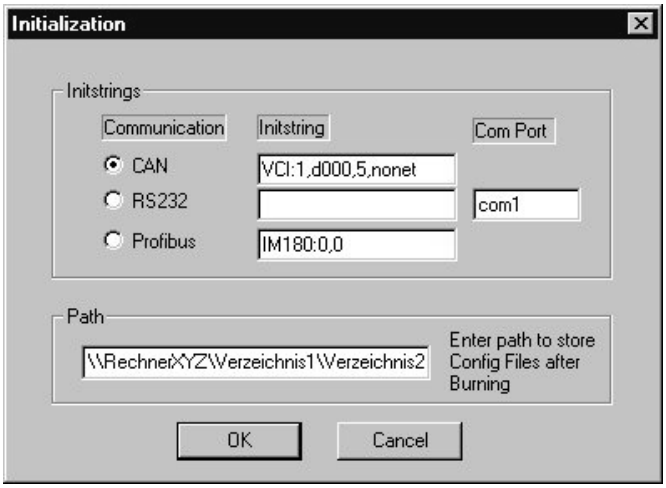
Dialog Language

Bislang noch nicht unterstützt, ist dieser Untermenüpunkt dafür vorgesehen, die Spracheinstellungen des Programms zwischen deutsch und englisch (später auch französisch) umschalten zu können.

6.7. Die Dialoge

Initialization

Dieser Dialog hat folgendes Aussehen:

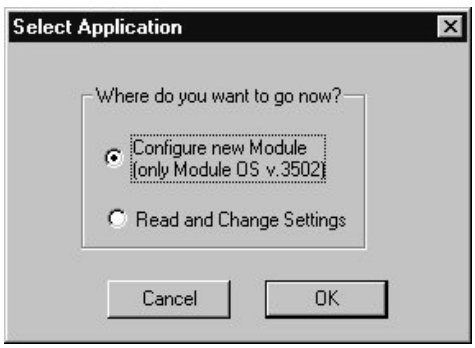


Er wird aufgerufen, um festzulegen, über welche Kommunikationsschnittstelle und mit welchem Initstring die Kommunikation mit dem Modul aufgebaut wird. Die Angabe des Com Port ist für die Kommunikation über die RS232-Schnittstelle wichtig, so auch, wenn z. B. über RS232 im sogenannten Bootstrapmode gebrannt werden soll. Die Pfadangabe bezieht sich auf den Dateipfad, in dem nach dem Brennen automatisch die Konfigurationsdatei des gebrannten Moduls abgelegt wird. Um dieses Feld wieder verlassen zu können, müssen Sie einen gültigen Pfad angeben, auf dem Sie Schreibrechte haben. Die Angabe eines Netzwerkpfades ist möglich. Der Dateiname der abgespeicherten Konfigurationsdatei setzt sich aus der Seriennummer und der Endung .cnf zusammen. Beim Betätigen des OK-Buttons werden die eingetragenen Einstellungen in der Ini-Datei PowerCfg.ini im Windows-Verzeichnis abgespeichert (siehe unter "Die Ini-Datei PowerCfg.ini") und sind sofort wirksam (z. B. beim Scannen nach Modulen). Mit Cancel werden alle Änderungen verworfen (Ausnahme: ungültige Pfadeingabe).

6.8. Die Verwendung des Konfigurationswizards

Ist der Menüpunkt "Use Wizard" aktiv (Häckchen gesetzt), so wird beim Aufruf des Menüpunkts "Edit Parameter" sowie nach erfolgreichem Scannen beim Doppelklick auf ein selektiertes Modul der Dialog "Select Application" geöffnet:

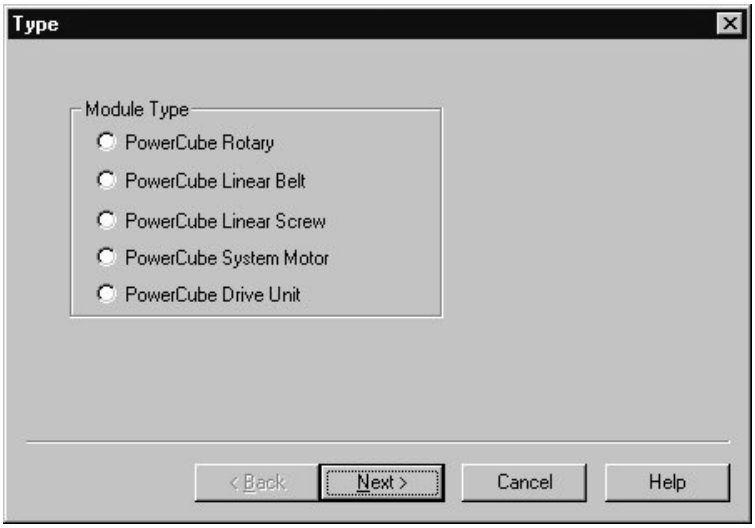
Select Application



Hier können Sie auswählen, ob Sie direkt weiter in den Hauptkonfigurationsdialog (Read and Change Settings) oder in einen speziellen Wizardmodus (Configure new Module) wechseln wollen. Dieser Wizardmodus ist nur für Module mit neuer Elektronik mit den Betriebssystemversionen 350X vorgesehen und kann daher nur bei Auswahl des Menüpunkts "Version 350X" genutzt werden! Bei Auswahl der Option "Configure new Module" und Betätigen des OK-Buttons gelangt man in einen Konfigura-

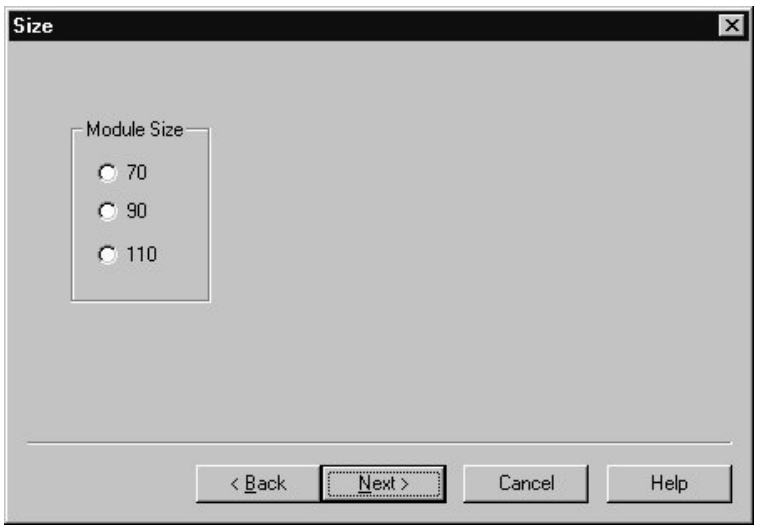
tionsdialog. Mit wenigen Mausklicks gestattet er es, die typischen Parameter für ausgewählte Modulklassen zu setzen.
Der erste Dialog dient zur Auswahl des Modultyps:

Type



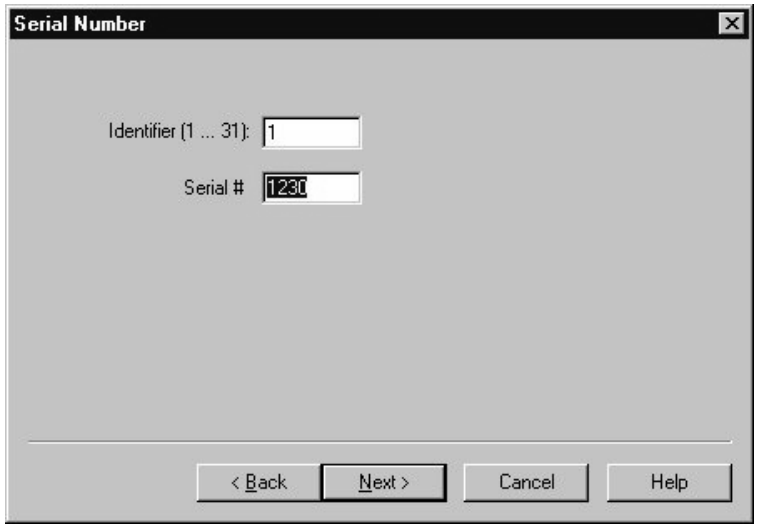
Nach Auswahl eines Modultyps kann mit Betätigen des Next Buttons in den nächsten Wizard-Dialog gewechselt werden.
Im zweiten Wizard-Dialog kann die Dimensionierung des ausgewählten Modultyps selektiert werden:

Size



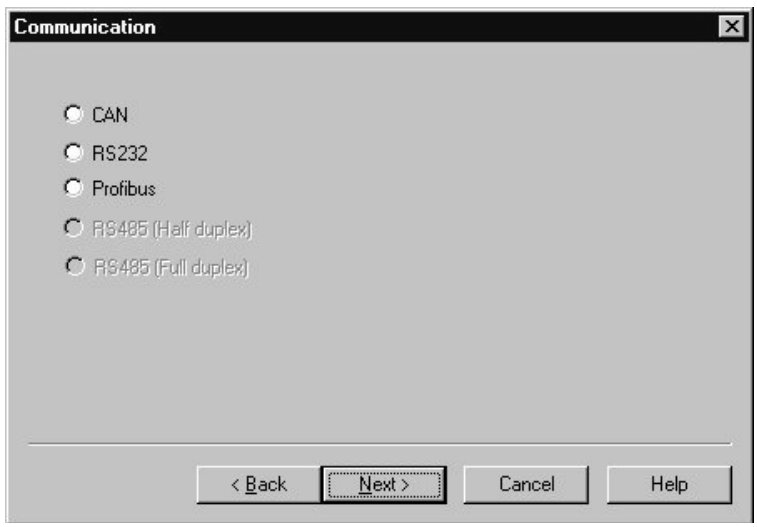
Nach Auswahl der Baugröße kann mit dem Next Button zum nächsten Wizard-Dialog gewechselt werden. Bitte beachten Sie, daß zur Zeit die Modul-Konfigurierung mit Hilfe des Wizards nur für PowerCube™-Drehmodule der Baugrößen 70 und 90 (benötigte Dateien: PR70_35.cnf und PR90_35.cnf) sowie für PowerCube™-Linearmodule mit Zahnriemen der Baugröße 70 (benötigte Datei: PLB70_35.cnf) unterstützt wird. Sollten diese Dateien nicht in Ihrem Lieferumfang enthalten sein, so können sie bei Bedarf via E-Mail von amtec bestellt werden. Zukünftige Versionen von PowerConfig werden gegebenenfalls auch weitere Modultypen und -baugrößen unterstützen. Über neue PowerConfig-Versionen können Sie sich auf der Support-Seite der amtec-Homepage, www.powercube.de, informieren.
Nach Auswahl eines unterstützten Modultyps und einer unterstützten Baugröße gelangen sie in den nächsten Wizard Dialog. Er bietet Ihnen die Möglichkeit der Auswahl einer Seriennummer und der Moduladresse (Modul-ID).

Serial Number



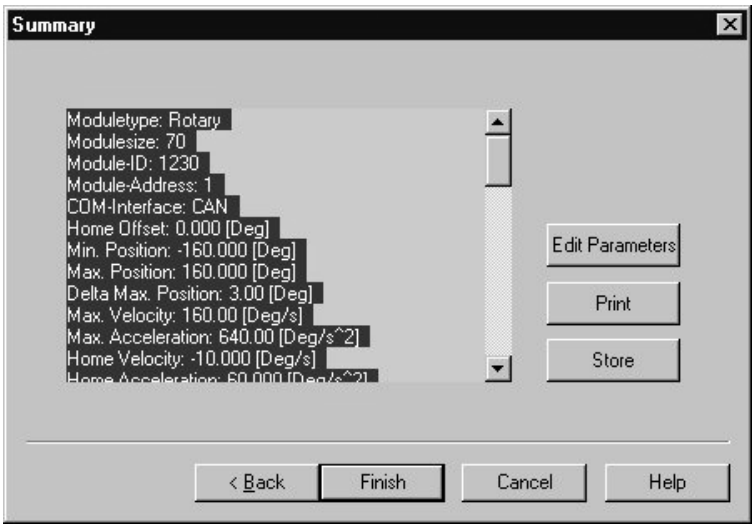
Es ist zu berücksichtigen, daß hierbei auch die Vergabe einer neuen Seriennummer möglich ist. Bei der Rekonfigurierung eines Moduls ist die auf dem Modul angegebene Seriennummer zu verwenden. Durch falsche Seriennummern wird der Support von Seiten amtec's extrem erschwert . Außerdem wird für Schäden, die aufgrund falsch eingebrannter Seriennummern entstehen, keine Haftung übernommen. Nach Eingabe der korrekten Daten kann mit dem Next Button zum nächsten Wizard-Dialog gewechselt werden:

Communication



In diesem Dialog kann die Kommunikationsschnittstelle festgelegt werden, über die mit dem Modul kommuniziert wird. Hierbei ist zu beachten, daß nicht nur über die Software das Kommunikationsinterface rekonfiguriert werden kann, sondern auch Hardwareänderungen am Modul vorgenommen werden müssen. Daher wird dringend abgeraten dem Modul eine andere Kommunikationsschnittstelle zuzuordnen, als es vor dem Brennen hatte, da sonst die Kommunikation nicht mehr hergestellt werden kann. Die Kommunikationsschnittstelle RS485 (Halbduplex oder Vollduplex) wird zur Zeit noch nicht unterstützt.
Nach dem Betätigen des Next Buttons gelangt man in den letzten Wizard-Dialog, in dem eine Zusammenfassung der neu zu brennenden Parameter angezeigt wird.
Diese setzen sich aus Ihrer Auswahl und der im Hintergrund geladenen Parameterdatei zusammen:

Summary



Bei Bedarf können hier Parameter noch verändert werden. Mit dem Back Button wird zu den gewünschten Wizard-Dialogseiten zurückgesprungen um dort neue Werte zu setzen. Weiterhin ist es durch Betätigen der Taste „Edit Parameter“ auch möglich in den Hauptkonfigurationsdialog zu springen und dort einzelne Parameter zu ändern. Mit den Tasten Print bzw. Store kann die so ausgewählte Modulkonfiguration ausgedruckt bzw. als ASCII-Datei gespeichert werden. Mit Cancel verlässt man den Konfigurationswizard und verwirft alle Einstellungen. Die Help-Funktion wird zur Zeit nicht unterstützt. Das Betätigen des Finish Buttons ruft den "Burning Module"-Dialog auf (siehe dort). An diesem Punkt können keine Parameteränderungen mehr vorgenommen werden. Mit Betätigen des Close Buttons werden alle Einstellungen verworfen.

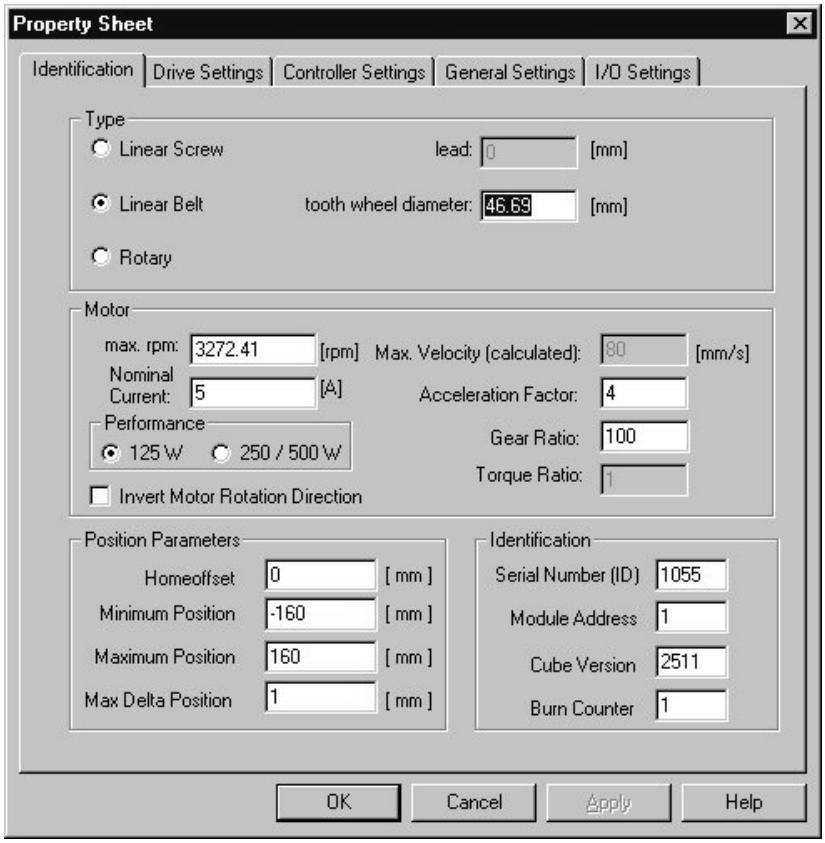
6.9. Modulparameter editieren

Der Hauptkonfigurationsdialog kann durch Anwahl des Untermenüpunkts "Edit Parameter“ nach Laden einer Konfigurationsdatei oder nach erfolgreichem Scannen nach Modulen auch durch Doppelklicken auf das gewünschte Modul aufgerufen werden.
Dieser Dialog unterteilt sich in drei Bereiche:

- Register für unterschiedliche Parametergruppen. Hinter jedem Register befindet sich ein Eigenschaftsfenster. Sie können zwischen den Registern hin und her schalten. Sie brauchen zuvor nicht den OK-Button klicken. Ihre Änderungen in den jeweiligen Registern bleiben erhalten.
- Der Bereich, in dem Sie die Eigenschaftenseite verlassen können. Bei einem Klick auf den Cancel-Button gehen Ihre Einstellungen verloren. Klicken Sie hingegen den OK-Button, so bleiben Ihre Einstellungen erhalten.
- Der dritte Bereich ist der wichtigste Bereich. Er sieht je nach Wahl des Registers anders aus. Es gibt insgesamt fünf Registerblätter.

Die erste, nach Aktivierung sichtbare Seite des Hauptkonfigurationsdialogs ist das Registerblatt "Identification“.

Identification



Folgende Parameter können hier gesetzt werden:

Type

Mit den Optionsbuttons können Sie den Modul-Typ angeben. Der korrespondierende Modulparameter wird in den nebenstehenden Eingabefeldern gesetzt.

- Lead: dieser Parameter wird für Linearmodule mit Spindeltrieb verwendet. Geben Sie die Spindelsteigung in Millimeter ein.
- Tooth wheel diameter: dieser Parameter wird nur für lineare PowerCube™-Module mit Zahnriemenantrieb verwendet. Geben Sie den Wirkdurchmesser der antreibenden Zahnscheibe in Millimeter ein.
- Rotary: der Faktor für Drehmodule ist immer gleich eins, daher ist kein Eingabefeld vorhanden.

Abhängig von Ihrer Auswahl werden alle Einheiten sofort umgestellt. Alle Einheiten für Linearmodule werden auf Millimeter umgestellt. Die Einheiten für Drehmodule werden auf Grad umgestellt. Bitte beachten Sie, daß keine Konvertierung der Werte stattfindet.

Max. rpm

Die maximal mögliche Drehzahl Ihres Motors. Diesen Wert entnehmen Sie bitte der Produktspezifikation. Dies ist eines der Felder, das zur Berechnung der empfohlenen Höchstgeschwindigkeit herangezogen wird. Die Eingabe eines neuen Wertes verändert die empfohlene Maximalgeschwindigkeit sofort. Hierbei ist zu beachten, daß die Eingabe eines unganzzahligen Wertes nur möglich ist, wenn erst die vollständige Zahl eingegeben wird und der Dezimalpunkt zum Schluß gesetzt wird.

Nominal Current

Der erforderliche Nominalstrom der Module beträgt 5A (Baugröße:70) bzw. 10A (Baugrößen 90 und 110). Dies entspricht den Einstellungen 15A bzw. 30A im Eingabefeld Max. Current der Vorgängersoftware easy*Config*.

Performance

Die Motorleistung. Die Eingabefelder für Nominalstrom und Motorleistung sind gekoppelt. Soll eine Motorleistung von 250/500W auf 125W geändert werden, dann muß der Nominalstrom vorher auf kleiner oder gleich 5A zurückgesetzt werden.

Max. Velocity (calc.)

Rechenfeld, das unter Berücksichtigung der max. Motordrehzahl (max. rpm), der Getriebeunter/übersetzung (Gear Ratio) und der Kenngröße für Linearmodule (Lead bzw. Tooth wheel diameter) die empfohlene Höchstgeschwindigkeit der Module berechnet. Die Berechnung erfolgt bei Änderungen im max. rpm-Feld sofort; bei Änderungen in den Feldern Gear Ratio, Lead und Tooth wheel diameter muß zur Neuberechnung in ein anderes Feld geklickt werden. Der berechnete

Geschwindigkeitswert gilt als maximaler Grenzwert und sollte keinesfalls überschritten werden. Das Ändern der Modulhöchstgeschwindigkeit darüber hinaus geschieht auf eigene Verantwortung!

Gear Ratio

Dieser Wert gibt die Getriebeübersetzung Ihres Antriebs an (z. B. 160 für eine Untersetzung von 160:1 oder 0,5 für eine Übersetzung von 1:2). Daher sollte keinesfalls der werkseitig voreingestellte Wert verändert werden. Ausnahme: Motion Controller und Systemmotoren, in die kundenseitig ein Getriebe eingebaut wird.

Torque Ratio

Dieser Wert ist immer auf "1" voreingestellt und kann derzeit nicht verändert werden.
Invert Motor Rotation Direction
Dreht die Motorlaufrichtung um. Dieser Wert ist ab Werk korrekt voreingestellt und darf nicht verändert werden. Nur unter der Modulbetriebsversion 350X zugänglich.

Homeoffset

Mit diesem Wert können Sie die "Position 0" des Moduls nach Ihren Wünschen konfigurieren, falls die Home position einmal nicht exakt mit dem von Ihnen gewünschtem Nullpunkt übereinstimmt.

Max./Min. Position

Diese beiden Parameter setzen die Bereichsgrenzen für auszuführende Bewegungen. Die Module akzeptieren im späteren Betrieb keine Werte, die außerhalb dieser Grenzen liegen.

Max Delta Position

Mit diesem Parameter setzen Sie die maximal erlaubte Abweichung der Ist-Position von der vom Modul berechneten Soll-Position. Befindet sich das Modul in Bewegung und wird diese Abweichung überschritten, so wird das Fehlerwort „TOW“ zurückgeliefert und ein „HALT“ ausgelöst.

Serial Number

Die Seriennummer des Moduls. Diese wird von amtec vergeben und darf nicht verändert werden (siehe auch "Der Wizard-Dialog Serial Number“).

Module Address

Dieser Parameter setzt die physikalische Adresse des Moduls und kann zwischen 1 und 31 liegen. Bitte beachten Sie, daß Verwendung gleicher Moduladressen in einem Verbund mehrerer Module zu Kommunikationsproblemen führt!

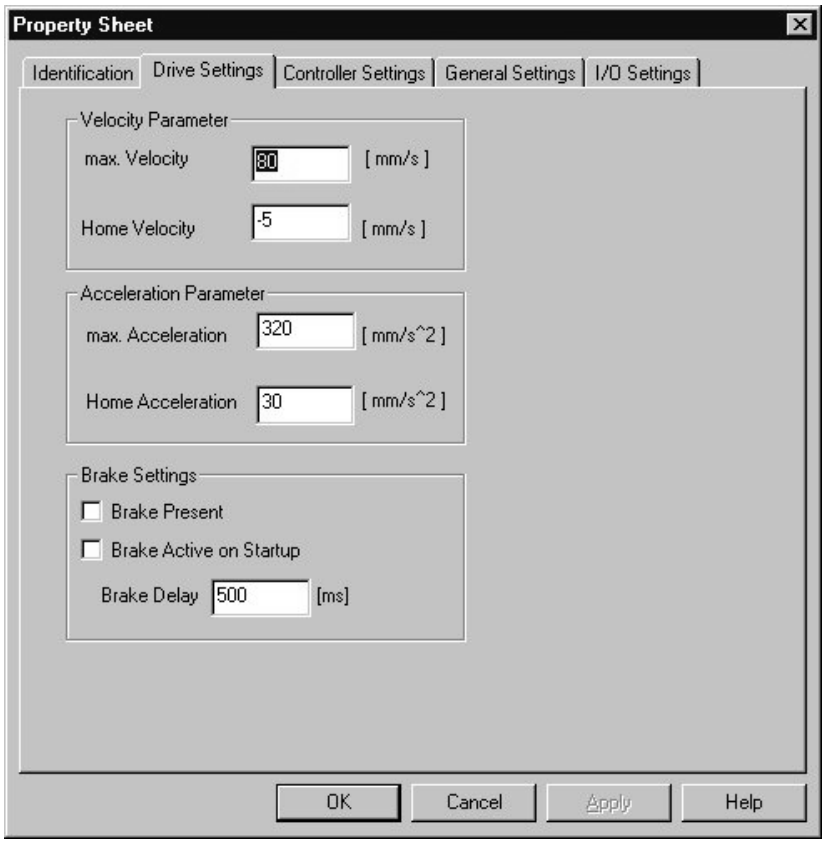
Cube Version

Dieser Wert zeigt die Version des Betriebssystems und kann nicht editiert werden.

Burn Counter

Zähler, der die Anzahl der am Modul durchgeführten Brennvorgänge protokolliert. Kann nicht editiert werden und wird zur Zeit nicht unterstützt.

Drive Settings



max. Velocity Dieser Parameter setzt die maximale Geschwindigkeit der PowerCube™-Module. Dia maximale Geschwindigkeit leitet sich aus der max. Drehzahl des antreibenden Motors und den Übersetzungsverhältnissen ab. Sie sollte den im Feld "Max. Velocity (calculated)" berechneten Wert nicht übersteigen. Die PowerCube™-Module akzeptieren im späteren Betrieb keine Werte, die größer sind als der hier eingestellte Wert.

Home Velocity Mit der hier eingestellten Geschwindigkeit wird die "Home position" angefahren. Dieser Wert kann auch vorzeichenbehaftet angegeben werden, um die Richtung der Referenzfahrt vorzugeben.

max. Acceleration Dieser Parameter setzt die maximale Beschleunigung der PowerCube™-Module. Die maximale Beschleunigung wird durch die Nennbeschleunigung des antreibenden Motors bestimmt. Die PowerCube™-Module akzeptieren im späteren Betrieb keine Werte, die größer sind als der hier eingestellte Wert.

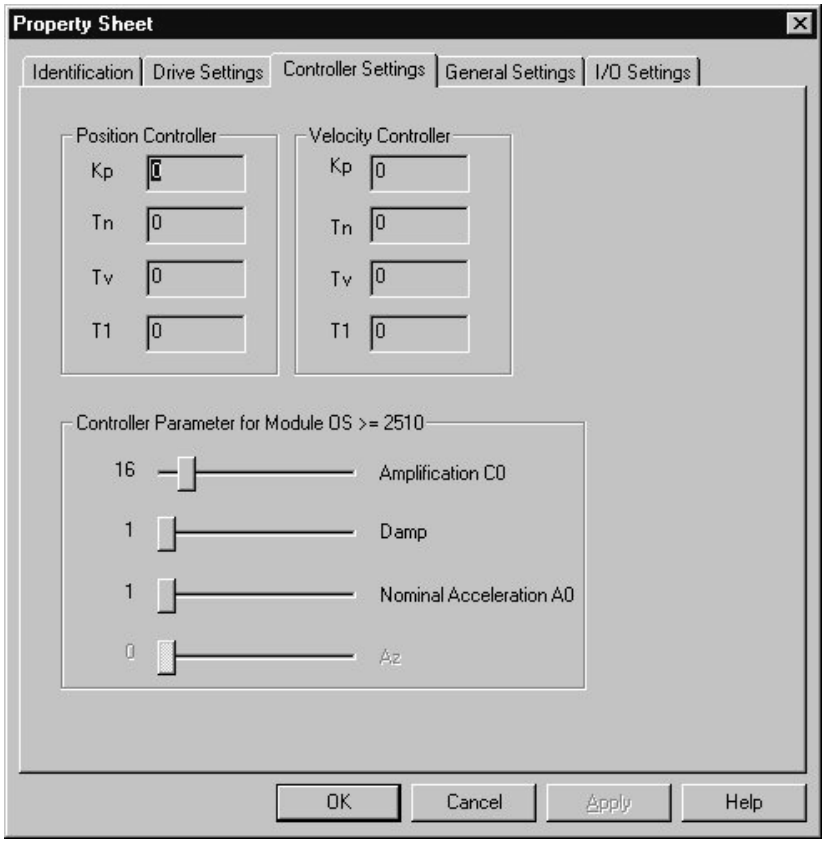
Home Acceleration Mit der hier eingestellten Beschleunigung wird die "Home position" angefahren.

Brake Present Bei Aktivierung dieser Eigenschaft schaltet der Antriebscontroller nach Abschluß der Bewegung automatisch die Bremse ein. Die Dauer zwischen Halt und Aktivierung der Bremse kann mit „Brake Delay" bestimmt werden (s. d.).

Brake Active at Startup Ist dieser Punkt aktiviert, so bleibt die Bremse nach dem Einschalten aktiv. Sie wird erst gelöst, wenn ein Fahrbefehl zur Ausführung kommt.

Brake Delay Dieser Parameter bestimmt die Dauer für die Zuschaltung der Haltebremse nach Beendigung des letzten Bewegungskommandos. Er sollte nicht kleiner als 500 ms und nicht größer als 5000 ms gewählt werden.

Controller Settings



Diese Seite ist nur für technisch versierte Personen gedacht. Hier können Sie die Regelkreisparameter im Mikroprozessor ändern. Je nach Betriebssystem steht Ihnen nur der obere Bereich (Version 250X) oder die unteren Schieberegler (Versionen 251X und 350X) zur Verfügung.
Diese Werte wurden werksseitig von amtec mit sinnvollen Werten voreingestellt und geprüft. Deshalb sollte ohne ernsthafte Notwendigkeit nichts verändert werden. Mit der m5dll wird das Programm "m5apitst.exe" mitgeliefert . Damit können zur Laufzeit des Programms die drei Reglerparameter C0, Damp und A0 neu gesetzt werden. Wurden

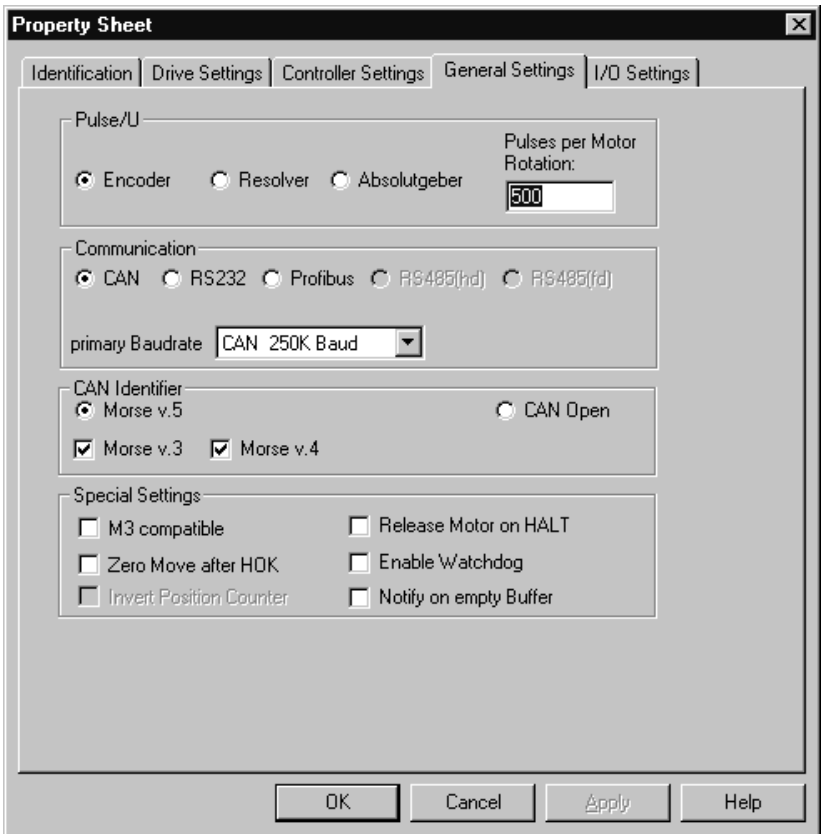
die passenden Werte gefunden, so wird m5apitst beendet und PowerConfig gestartet, ohne die Stromversorgung des Moduls abzuschalten. Die neuen Reglerparameter werden nun direkt von PowerConfig übernommen und können sofort gebrannt werden. Erfolgt kein Einbrennen der neuen Controllerparameter in das Modul, gehen sie beim Abschalten der Stromversorgung des Moduls verloren.

Amplification C0. Dieser Wert stellt einen Verstärkungsfaktor dar. Je größer C0 gesetzt wird, desto geringer wird die Verstärkung.

Damp Dieser Wert stellt einen Dämpfungsfaktor dar. Je größer Damp gesetzt wird, desto größer wird die Dämpfung.

Nominal Acceleration A0. Hier wird die Beschleunigungsfähigkeit des Antriebs eingestellt. Je größer A0 gesetzt wird, desto größere Beschleunigungsfähigkeit wird unterstellt.
Az. Die Änderung dieses Wertes wird nicht unterstützt.

General Settings



Feedback Die Rückführung der Lageinformation kann über:

- Encoder Feedback
- Resolver Feedback
- Absolute Feedback

realisiert werden. Auch diese Werte haben direkten Einfluß auf die Hardware der PowerCube™-Module. Diese Werte sollten nicht geändert werden, wenn nicht auch die Hardware modifiziert wurde.

Pulses Per Rotation Dieser Parameter beschreibt die Anzahl der Impulse je Umdrehung der Encoder-Scheibe im Modul. Eine Änderung dieses Parameters sollte mit Vorsicht geschehen, da er mit der Hardware im PowerCube™-Modul zusammenhängt.
Communication
Die Kommunikation kann über folgende Schnittstellen erfolgen:

- CAN
- RS232
- Profibus (nur Modulbetriebsversion 350X)

Das Modul muß hardwareseitig für die unterschiedlichen Schnittstellen vorbereitet sein. Die werksseitig vorgenommene Voreinstellung sollte daher nicht verändert werden, ohne daß auch entsprechende Veränderungen an der Hardware vorgenommen wurden. Wird dies nicht beachtet, ist die Kommunikation mit dem Modul

nicht mehr möglich! Die Schnittstelle RS485 (Halbduplex sowie Vollduplex) wird zur Zeit nicht unterstützt.

Primary Baudrate In diesem Popup-Menü können je nach Auswahl der Kommunikationsschnittstelle die Übertragungsfrequenz der CAN- bzw. der RS232-Kommunikation einstellt werden. Für die Profibusschnittstelle sind keine Einstellungen erforderlich.

CAN Identifier Diese Optionen sind nur unter der Modulbetriebsversion 350X zugänglich. Hier erfolgt eine Auswahl der CAN-Identifier, auf die das Modul antworten soll. Bei Auswahl des Morse v.5-Modus kann darüber hinaus festgelegt werden, ob das Modul Morse v.3 und/oder Morse v.4 kompatible Befehle verstehen soll.

M3 Compatible Das Modul wird in den vollständigen Morse v.3-Kompatibilitätsmodus geschaltet. Im Gegensatz zur Einstellung Morse v.3 bei den CAN-Identifiern versteht das Modul nicht nur Morse v.3-Befehle, sondern verhält sich bei Initialisierung mit einem entsprechenden Initstring vollständig wie ein Morse v.3-Modul. Dies bedeutet:

- automatisches Senden des Modulstatus nach Aktivierung durch CANID-Modulrequest.
- Verweigerung von Bewegungsbefehlen, solange keine Referenzpunktfahrt erfolgt ist.

Zero Move after HOK Diese Option ist nur unter der Modulbetriebsversion 350X zugänglich. Das Modul wird direkt nach der Referenzfahrt auf die benutzerdefinierte Nullposition bewegt (siehe Homeoffset).

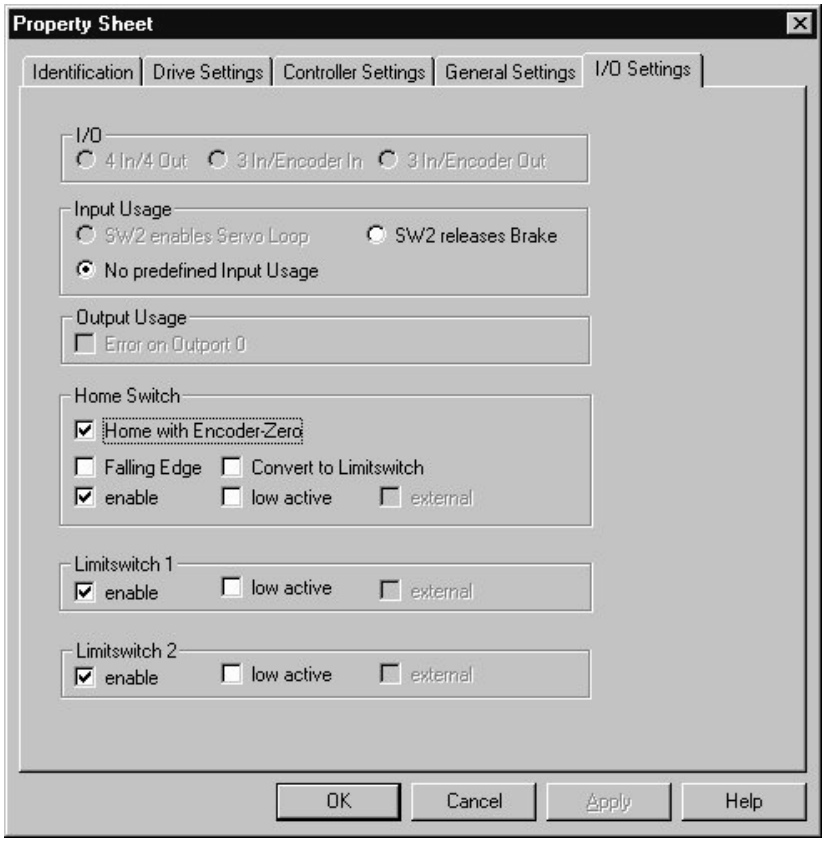
Invert Position Counter Diese Option ist nur unter der Modulbetriebsversion 350X zugänglich. Die Vorzeichen der min./max. Position können vertauscht werden. Wird zur Zeit nicht unterstützt.

Release Motor on HALT Diese Option ist nur unter der Modulbetriebsversion 350X zugänglich. Wird per Nothalt oder in Kombination mit einem Fehlerstatus das HALT-Flag gesetzt, so wird die Motorregelung abgeschaltet.

Enable Watchdog Diese Option ist nur unter der Modulbetriebsversion 350X zugänglich. Die im Modul implementierte Watchdog-Funktion wird freigeschaltet. Vom Anwendungsprogramm kann zugleich ein „Lebenszeichen“ unter Nutzung eines CANID_CMDALL gesendet werden. Wurde das „Lebenszeichen“ einmal gesendet, so muß es spätestens nach 100 ms wiederholt werden. Anderenfalls löst das Modul selbständig einen Nothalt aus. Damit wird vom Modul überprüft, ob sich der Steuerrechner noch im normalen Arbeitsmodus befindet oder eine Fehlfunktion hat. In letzterem Fall schaltet sich das Modul ab.

Notify on empty Buffer Diese Option ist nur unter der Modulbetriebsversion 350X zugänglich. Eine Benachrichtigung wird geschickt, wenn im Step-Bewegungsmodus der Befehlspuffer leer ist.

I/O Settings



I/O

Diese Einstellungen sind nur unter der Modulbetriebsversion 350X zugänglich. Hier wird die Bedeutung der Input/Output-Signale festgelegt, die am 15-poligen I/O-Interface anliegen (siehe Pinning des zusätzlichen 15-poligen I/O-Interfaces).
Zu beachten ist, daß die Veränderung dieser Einstellung mit H/W-Einstellungen im Modul zusammenläuft und auch die Bedeutung der Pins beeinflußt . Einer Änderung der I/O-Settings muß eine Umschaltung der H/W vorausgegangen sein.

Input Usage

Diese Einstellung bestimmt, welches Modulverhalten ein Eingangssignal auf dem für den externen Endschalter 2 bestimmten Datenkanal auslöst (siehe Pinning des zusätzlichen 15-poligen I/O-Interfaces).

- SW2 enables Servo Loop: Diese Option ist nur unter der Modulbetriebsversion 350X zugänglich. Ein High auf der SW2-Datenleitung aktiviert die Motorregelung. Ein Low deaktiviert die Motorregelung.
- SW2 releases Brake: Ein High auf der SW2-Datenleitung löst die Bremse. Ein Low verriegelt die Bremse.
- No predefined Input Usage: Ein Eingangssignal auf der SW2-Datenleitung löst kein vordefiniertes Modulverhalten aus.

Diese Einstellungen können nur getroffen werden, wenn kein externer Endlagenschalter 2 an das Modul angeschlossen ist! Daher sind die betreffenden Optionsfelder bei aktiviertem externen Limitswitch 2 (s. u.) nicht zugänglich.

- Output Usage: Error on Outport 0

Bei Aktivierung dieser Option wird im Fehlerfall vom Modul ein Signal über den Datenausgangskanal 0 gesendet (siehe Pinning des zusätzlichen 15-poligen I/O-Interfaces).

Home Switch

Der Referenzschalter wird vom Betriebssystem der PowerCube™-Module verwendet, um während einer Homing-Prozedur eine absolute Nullage zu finden.

- Home with Encoder-Zero: Um die Initialisierungsprozedur mit dem Referenzschalter zu triggern und sie mit dem "Zero-Impuls" des Encoders zu beenden, ist dieser Punkt zu markieren. Anderenfalls wird die Schaltfläche des Referenzschalters allein für die Referenzfahrt genutzt.
- Falling Edge: Der Schalter arbeitet auf der fallenden Flanke, bei

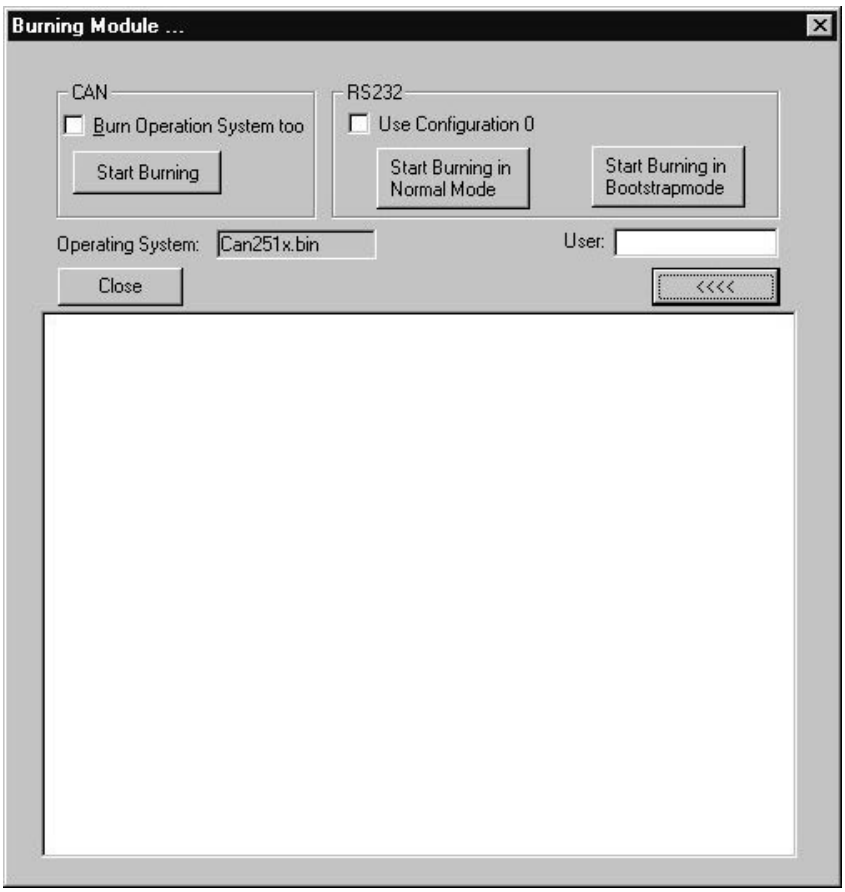
Nichtmarkierung auf der steigenden Flanke.

- Convert to Limit Switch: Nach Durchführung einer Homing-Prozedur ändert sich das Verhalten des Referenzschalters in das eines Endlagenschalters.
- Enable: Aktivierung der Referenzschalter. Wenn dieser Punkt nicht aktiviert ist, werden die übrigen Einstellungen der Referenzschalter ignoriert.
- Low active: Der Schalter ist low-aktiv, bei Nichtmarkierung high-aktiv.
- External: Es wird der externe Referenzschalter-Eingang ausgewertet.

Limitswitch 1 und 2

- Enable: Schalter aktiv.
- Low active: Der Schalter ist low-aktiv, im anderen Fall high-aktiv.
- External: Es werden die externen Endschaltereingänge ausgewertet (siehe Pinning des zusätzlichen 15-poligen I/O-Interfaces). Im Falle des Limitswitch 2 (SW2) wird mit Setzen dieser Option die Sektion “Input Usage“ deaktiviert (s. o.).

Burning Module



In diesem Dialog-Fenster werden schließlich die ausgewählten Einstellungen durch Flash-Download der Daten in ein EPROM im Modul fixiert. Es gibt zwei Schnittstellen, über die Daten in das Modul gebrannt werden können:

Flash per CAN

Der Button „Start Burning“ wird für den Fall verwendet, daß die PowerCube™-Module per CAN-Interface angeschlossen sind. Sie haben nun zwei Möglichkeiten. Der erste Fall ist der Normalfall, indem nur die Parameter in das Modul gebrannt werden. Wird hingegen die Auswahl "Burn Operation System too" getroffen und anschließend "Start Burning", werden nicht nur die Parameter in das Modul gebrannt, sondern auch das Betriebssystem. Die Auswahl des Betriebssystems findet automatisch durch die Selektion der Version im Menü "Hardware" sowie der Kommunikationsschnittstelle im Konfigurationsdialog "General Settings" statt. Ein Upgrading vom Betriebssystem 250X auf 251X kann durch Selektieren von „Version 251X“ im Menü „Hardware“ nach dem Scannen des betreffenden Moduls erreicht werden. Es ist zu beachten, daß dabei Controller-Parameter neu gesetzt werden müssen. Ein umgekehrtes Downgrading wird nicht empfohlen! Die notwendigen Betriebssytemdateien sind im Verzeichnis „OS

Directory“ von PowerConfig zu finden. Im Falle der CAN-Kommunikation sind das "can250x.bin", "can251x.bin" sowie "can350x.bin"; für die RS232-Schnittstelle existieren "Rs250x.bin", "Rs251x.bin" und "Rs350x.bin". Die Betriebssystemversionen für die Profibusschnittstelle sind "Pbus250x.bin", "Pbus251x.bin" sowie "Pbus350x.bin". In der PowerConfig.ini Datei sollte der Pfad, in dem die Betriebssystemdateien liegen, unter dem Punkt „OS Dirctory=“ in der Sektion [Software] eingetragen oder die Dateien in das Arbeitsverzeichnis von PowerConfig kopiert werden, da das Programm diese Dateien sonst nicht findet.

Flash per RS232

Der Button "Start Burning in Bootstrapmode" in diesem Bereich wird verwendet, um das Brennen über ein spezielles Bootstrap-Kabel zu ermöglichen. (Dieses Bootstrap-Kabel ist gesondert bei amtec zu ordern.) Dabei wird das Betriebssystem des Moduls immer neu gebrannt. Wird die Option "Use Configuration 0" selektiert, kann zusätzlich ein geladener Parametersatz in das Modul gebrannt werden. Diese Möglichkeit ist z. B. dann hilfreich, wenn keine Kommunikation im normalen CAN-Betrieb mehr möglich ist. Hierzu muß das Bootstrap-Kabel mit einem speziellen Stecker im Modul verbunden werden, der nach Abschrauben der 15-poligen I/O-Buchse zugänglich wird. Unterstützung für die Nutzung dieser Option erhalten Sie von Ihrem Servicepartner .

Die Benutzung des Buttons "Start Burning in Normal Mode", der das Brennen über ein normales RS232-Kabel ermöglichen soll, wird zur Zeit noch nicht unterstützt.
Bitte berücksichtigen Sie, daß sich höchstens zwei konfigurierbare Bewegungsachsen an der Datenleitung befinden dürfen, damit der Brennvorgang in Gang gesetzt werden kann (PowerCube™ Handgelenksmodule (PW) sowie Lift'n Turn-Module (PLT)). Anderenfalls wird eine Fehlermeldung ausgegeben, ohne das eine Brennvorgang durchgeführt wurde.
Das Feld "Operating System“ zeigt die Betriebssystemversion an, die bei Bedarf in das Modul-EPROM gebrannt wird. Es kann nicht editiert werden.
Das Feld "User“ muß vor Beginn des Brennvorgangs mit mindestens einem Zeichen gefüllt sein. Der Mitarbeitername bzw. das -kürzel werden beim Ausdrucken und Speichern der Konfigurationsdatei als ASCII-Datei mit ausgegeben.
Beim Starten des Brennprozeßes, ist eine Bestätigung erforderlich. Fällt die Bestätigung positiv aus, wird der Brennvorgang sofort eingeleitet. Am Ende dieses Vorganges erfolgt eine Nachricht über Erfolg oder Mißerfolg. Im Falle eines Mißerfolges kann der Versuch unternommen werden, den Vorgang erneut zu starten Im Normalfall wird das nicht gelingen, da das Modul zu keiner Kommunikation mehr fähig ist. Die Versorgungsspannung ist einmal aus- und wieder einzuschaltenund ein erneuter Vorgang kann eingeleitet werden.
Wenn der Brennvorgang erfolgreich war, ist die Versorgungsspannung einmal aus- und wieder einzuschalten. Damit lädt das Modul die neuen Vorgabewerte in das RAM. Beim Schließen des Dialogs via Close-Button bleiben alle geänderten Parameter dem selektierten Modul zugeordnet, sind aber nicht physikalisch ins Modul gebrannt. Beim Schließen des Programms sind dann alle Änderungen verworfen.

6.10. Fehlerbehandlung

ACHTUNG: Zum Brennen von Modulen sollte PowerConfig nur auf Rechnern mit CPU’s ab Pentium 60 benutzt werden. Die Benutzung von Rechnern mit 486er-CPU zum Konfigurieren von Modulen zieht mit hoher Wahrscheinlichkeit Brennfehler nach sich, die ein erneutes Brennen im RS232-Bootstrapmode nötig machen. Für Schäden, die aufgrund falscher Rechnerkonfiguration beim Brennen am Modul entstehen, übernimmt amtec keine Haftung.
Bitte unbedingt beachten: Beim Brennen mit PowerConfig muß in jedem Fall der korrekte Interrupt angegeben werden.
ISA-CAN-Karten der Firma IXXAT: Der 3. Parameter des Initstring enthält den korrekten Interrupt. Beispiel: VCI:1,d000,5
PCI-CAN-Karten der Firma IXXAT: Der 3. Parameter des Initstring muß eine Zahl größer Null enthalten. Dann wird der vom Betriebssystem für diese Karte reservierte Interrupt verwendet. Beispiel: VCI:6,0,1
CANdy-Karte: Unter Win95/98 Anschluß der Karte an den LPT-Port. Im Initialisierungsstring muß der IRQ des LPT-Ports angegeben werden, da der

Flashdownload sonst fehlschlägt! Unter WinNT/2000 gibt es derzeit keine Möglichkeit, über die CANdy-Karte Module zu brennen.

Folgende bekannte Schwierigkeiten können bei der Installation bzw. der Benutzung des Programms auftreten:
Unter Windows NT wird die Installation mit der Meldung „Installation incomplete“ abgebrochen. Lösung: Die Installation ist als Administrator durchzuführen!
Unter Windows 9x stürzt das Programm nach Eingabe des Initstrings und Scannen nach Modulen mit einer Schutzverletzung der Kernel32.dll ab. Lösung: Im Initstring ist immer die Option nonet anzugeben (siehe: "Der Aufbau des Basis-Initialisierungsstrings).
Die Seite „Controller Settings“ kann nicht angezeigt werden. Lösung: Installation des Internetexplorer ab Version 4.01 => Update des Betriebssystems.
Beim Versuch, ein Modul zu brennen, kommt in der Statusanzeige des Brenndialoges die Fehlermeldung: „Error sending Flash Mode Command! [207]“. Lösung: Im Initstring muß explizit der Interrupt der Schnittstelle angegeben werden (bei PCI-Karten: eine Zahl ungleich Null; zeigt an, daß Interrupt benutzt werden soll). Das Modul hat dabei keinen Schaden genommen; nach Aus- und Einschalten des Stroms ist es wieder normal ansprechbar.
Beim Versuch, ein Modul zu brennen, kommen in der Statusanzeige des Brenndialoges die Fehlermeldungen: „Cannot erase bank ...! But I try to burn it...“ und „CheckClear failed“. Lösung: Im Initstring muß explizit der Interrupt der Schnittstelle angegeben werden (bei PCI-Karten: eine Zahl ungleich Null; zeigt an, daß Interrupt benutzt werden soll). Diese Meldung kann auch erscheinen, wenn das Programm auf einem Rechner mit 486-CPU ausgeführt wurde. ACHTUNG: Beim Auftreten dieser Fehlermeldungen wurden Speicherbänke des Moduls überschrieben. Es ist nun nicht mehr ansprechbar und muß über den RS232-Bootstrap-Mode mit einer neuen Ur-Konfiguration versehen werden. Dazu ist unbedingt der amtec-Kundenservice zu konsultieren!

7. Kommunikation

Das Betriebssystem jedes PowerCube™-Antriebs erfüllt folgende Aufgaben:

- Echtzeitsteuerung und -regelung des Motors auf Grundlage der Lageerfassung.
- Überwachung der Stromaufnahme des Motors und der Temperatur der Motorwicklung und Endstufentransistoren sowie der Endlagen des Systems.
- Ausführung der Bewegungsfunktionen nach den Kommandos der übergeordneten Steuerung.
- Steuerung der Bremse (sofern eine Bremse im Antrieb integriert ist).

In den folgenden Abschnitten sind die Funktionen des Betriebssystems näher erläutert. Dazu gehören Informationen zum Datenaustausch zwischen Master und Slave sowie über Diagnostizierung und Behebung von Fehlermeldungen des PowerCube™.

7.1. Kommunikationsschnittstellen

Die PowerCube™-Antriebsmodule der amtec GmbH werden mittels einer seriellen Kommunikationsschnittstelle angesteuert, über die alle Kommandos zur Bewegungssteuerung, Parametrierung und Überwachung zwischen den Modulen und der übergeordneten Steuerung ausgetauscht werden.
Die Kommunikationsschnittstelle ist gegenwärtig in Form eines CAN-Feldbusprotokolls (nach ISO/DIS 11898), RS232, Profibus sowie auf RS-485 realisiert.

7.2. PowerCube™ Kommunikationsprotokoll

Für die Übertragung von Daten zwischen Master und PowerCube™ wird unabhängig von der verwendeten Busschnittstelle ein einheitliches Datenprotokoll eingesetzt. Die in serieller Weise übertragenen Daten werden vom Antrieb empfangen, interpretiert und beantwortet. Die Dauer für diesen Prozeß ist von der eigentlichen Busschnittstelle und den gewählten Übertragungsparametern abhängig.
Ein PowerCube™-Kommando hat folgenden grundsätzlichen Aufbau:

Identifier	CommandID	ParameterID	Data 0 ... Data 7

Tabelle 27

Die übertragenen Command ID's, Parameter ID's und Folgebytes sind für alle Kommunikationsarten (CAN, RS232, Profibus-DP) der PowerCube™-Module identisch. Die Identifier variieren entsprechend der Vorgaben des eingesetzten Busprotokolls.

7.3. PowerCube™-Identifier für den CAN-Bus

Die im PowerCube™ verwendeten CAN-Identifier entsprechen der CAN-Spezifikation 2.0 Part A mit 11 Identifier-Bits

CAN-Identifier	1 LSB	1	2	3	4	5	6	7	8	9	10	Len	Hexadezi-mal-wert

Tabelle 28

CANID_CMDACK	m	m	m	m	m	1	0	1	X	X	X	1-8	0x0a0 + Modul-ID
CANID_CMDGET	m	m	m	m	m	0	1	1	X	X	X	2	0x0c0 + Modul-ID
CANID_CMDPUT	m	m	m	m	m	1	1	1	X	X	X	6-8	0x0e0 + Modul-ID

Tabelle 28

m = Module-ID, X = reserviert für interne Verwendung
Ein spezieller CAN-Identifier ermöglicht die Kommunikation mit allen angeschlossenen Modulen über eine einzige Kommandozeile:.

CAN-Identifier	0 LSB	1	2	3	4	5	6	7	8	9	10	Len	Hexadezimal-wert
CANID_CMDALL	0	0	0	0	0	0	0	0	1	0	0	1/8	0x100

Tabelle 29

Die CAN-Identifier 0x3e9 bis 0x7e5 sind für die Anwendung von bis zu 51 I/O-Modulen der Firma Thomas Wünsche reserviert.

7.4. PowerCube™-Identifier für den Profibus

Die Adressierung der Teilnehmer im Profibus-Protokoll ist Bestandteil der physikalischen Schicht. Es ist daher kein Identifier für Profibus-Anwender erforderlich. Die PowerCube™ Module wurden von der PNO (Profibus- Nutzer- Organisation) zertifiziert und sind somit an allen Standard- Profibus DP- Mastern betriebsfähig. Gleichzeitig können sie mit beliebigen DP- Slaves fremder Hersteller kombiniert betrieben werden. Voraussetzung dafür ist die korrekte Konfiguration des DP- Masters. Zu diesem Zweck ist die mitgelieferte GSD- Datei (Cube1641.gsd) in das Konfigurationsprogramm einzulesen. Ein PowerCube™ verfügt über 8 Ausgangs- Bytes und 16 Eingangs- Bytes.
Die Profibus- Kommunikation erfordert folgende wichtige Maßnahme zum störungsfreien Betrieb der PowerCube™ Module:
Um das Modul zu veranlassen, ein neues Kommando zu bearbeiten (z. B. Pollen von Status oder Position), muß sich mindestens eines der 8 Ausgangs- Bytes beim Senden ändern. Dies läßt sich einfach lösen, indem das nicht verwendete Byte 7 (Zählung beginnt bei Null) wie ein Zähler inkrementiert wird.
Organisation der 16 Eingangs- Bytes
Die unteren 8 Eingangs- Bytes enthalten stets die Antworten des Moduls auf die gestellten Anfragen. Der Inhalt wird in den nachfolgenden Abschnitten des Handbuchs genauer beschrieben.
Die oberen 8 Eingangs- Bytes sind profibus- spezifisch und enthalten bei jeder Antwort zusätzlich folgende Informationen:

IN Byte8	**IN Byte9**	**IN Byte10**	**IN Byte11**	**IN Byte12**	**IN Byte13**	**IN Byte14**	**IN Byte15**
-	----------Int32 Position [Ink]----------				-----Kurzstatus-----		ms (0-255)

Tabelle 30

Der Profibus DP Kurz- Status ist wie folgt definiert:

Profibus DP Kurz-Status	**Wert**	**Gleichbedeutend wie**
STATE_ERRORM4	0x0001	STATE_ERROR

Tabelle 31

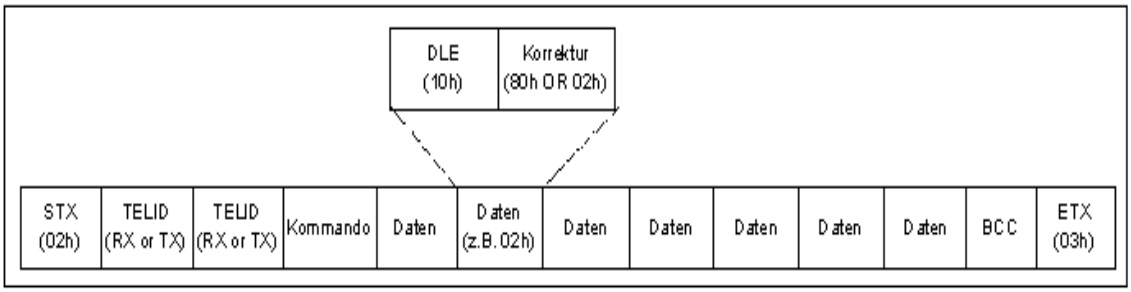
STATE_READYTOPOS	0x0004	STATE_HOME_OK
STATE_MOVING	0x0010	STATE_MOTION
STATE_BRAKED	0x0020	STATE_BRAKEACTIVE
STATE_CURRLIMIT	0x0040	STATE_CURLIMIT
STATE_ENDMOTION	0x0080	STATE_RAMP_END
STATE_END0	0x0100	STATE_SW1
STATE_END1	0x0200	STATE_SW2
STATE_SYNC	0x0400	STATE_SWR
POSTIME_INPROGRESS_MASK	0x1000	STATE_INPROGRESS
POSTIME_RECEIVED_MASK	0x2000	STATE_FULLBUFFER
STATE_HALTM4	0x8000	STATE_HALTED

Tabelle 31

Wertet man die oberen 8 Eingangs- Bytes entsprechend aus, erübrigt sich das separate Pollen von Status und Position. Darüber hinaus kann die kontinuierliche Änderung der ms- Information (Millisekunden) als "Lebenszeichen" des Moduls (watchdog) interpretiert werden.

7.5. PowerCube™-Identifier für die RS232-Kommunikation

Der Datenverkehr via RS-Kommunikation wird nicht wie bei den industriellen Feldbussystemen (CAN, Profibus) durch eine Hardware-Schicht unterstützt, die eine fehlerfreie Datenübertragung garantiert. Aus diesem Grunde hat amtec ein eigenes Datenprotokoll zur sicheren Übertragung von Daten über die RS-Schnittstellen entworfen.
Die Telegrammstruktur unter RS232 hat folgendes Aussehen:



Ein Frame wird also immer von dem Sonderzeichen STX (02h) eingeleitet und mit dem Sonderzeichen ETX (03h) abgeschlossen. Kommen innerhalb des Telegrammes diese Sonderzeichen vor, so werden Sie durch die Zeichenkombination DLE (10h) und (80h OR Zeichen) ersetzt. Auch das hierbei verwendete Sonderzeichen DLE (10h) wird auf die gleiche Weise ersetzt, so daß sich folgendes Substitutionsschema ergibt:

- 0x02 wird zu 0x10 0x82
- 0x03 wird zu 0x10 0x83
- 0x10 wird zu 0x10 0x90.

Diese DLE-Korrektur gilt für den gesamten Datenbereich. Die Telegrammlänge kann also maximal 24 Byte betragen.
Als BCC wird der Block Check Character bezeichnet, eine Prüfsumme berechnet aus der Summe über alle Nettodaten (ohne Berücksichtigung der DLE-Korrektur). Die Prüfsicherheit beträgt mit dieser Methode ca. 95%.
BCC = (TELID + Command + Data);
BCC = BCC + (BCC >> 8);

Der 16-Bit Identifier TELID unterscheidet Befehle an das Modul (TELID_SENDDAT) und Antworten vom Modul (TELID_RECVDAT). Sie haben folgenden Aufbau:

	1. Byte								2. Byte							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TELID_SENDDAT	0	0	0	0	0	1	m	m	m	m	m	0	n	n	n	n
TELID_RECVDAT	0	0	0	0	1	0	m	m	m	m	m	0	n	n	n	n

Tabelle 32

m: Modul ID 0 ... 31
n: Anzahl Folgebytes (netto)
Eine Nachricht wird dann vom Modul interpretiert, wenn

- das Sende ID in TELID steckt,
- die Moduladresse stimmt und
- die Anzahl der empfangenen Netto-Daten mit der kodierten Länge übereinstimmt.

Die Prüfsumme wird aus Kompatibilitätsgründen derzeit nicht ausgewertet.

7.6. Übersicht über die PowerCube™-Command ID's

Im folgenden erhalten Sie einen Überblick darüber, in welcher Reihenfolge die notwendigen Daten an den PowerCube™ zu senden sind. Anhand der übertragenen Daten wird zunächst festgelegt, welches Kommando abgearbeitet werden soll. Dabei sind folgende Kommandos verfügbar:

Kommando	Command ID	Länge	Bedeutung
Reset	0x00	1 Byte	Fehlerstatus löschen
Home	0x01	1 Byte	Referenzfahrt auslösen
Halt	0x02	1 Byte	Nothalt auslösen
RecalcPIDParam	0x09	1 Byte	Neuberechnen der Regelschleife
SetExtended	0x08	3-6 Byte	Parameter setzen
GetExtended	0x0a	2 Byte	Parameter lesen
SetMotion	0x0b	6-8 Byte	Bewegung ausführen
SetIStep	0x0d	7 Byte	Step-Bewegungskommando

Tabelle 33

Die gesendeten Daten beantwortet das Modul, sofern dies im Konfigurationswort Config festgelegt ist, mit einem Acknowledge, so daß der Datenfluß der Informationsbytes wie nachfolgend beschrieben zusammengefasst werden kann (siehe auch Kap. Aufbau der Datenströme):

Kommando	Send	Acknowledge
Reset	[0x00]	[0x00]
Home	[0x01]	[0x01]
Halt	[0x02]	[0x02]
RecalcPIDParam	[0x09]	[0x09]
SetExtended	[0x08] [param] [data] ...	[0x08] [param] [0x64]
GetExtended	[0x0a] [param]	[0x0a] [param] [data] ...

Tabelle 34

SetMotion	[0x0b] [mode] [data] ...	[0x0b] [mode] [0x64]
SetIStep	[0x0d] [data] ...	[0x0d]

Tabelle 34

Für die Kommandos „Parameter lesen“ (GetExtended) und „Parameter schreiben“ (SetExtended) ist im nachfolgenden Datenbyte festgelegt, auf welchen Parameter sich die zu übertragenden Daten beziehen. Die Anzahl der sich daran anschließenden Datenbytes ist abhängig vom Datentyp des betreffenden Parameters. Die Liste der Parameter ist sehr umfangreich und wird im folgenden näher erläutert.
Das Kommando „Bewegung ausführen“ bedingt als Folgewert den Bewegungsmodus. Die Anzahl der darauf folgenden Datenbytes richtet sich nach dem gewählten Bewegungsmodus. Eine Tabelle im Abschnitt 7.2.8 zeigt die verschiedenen Bewegungsarten und die zugehörigen Parameter.

7.7. Datentypen

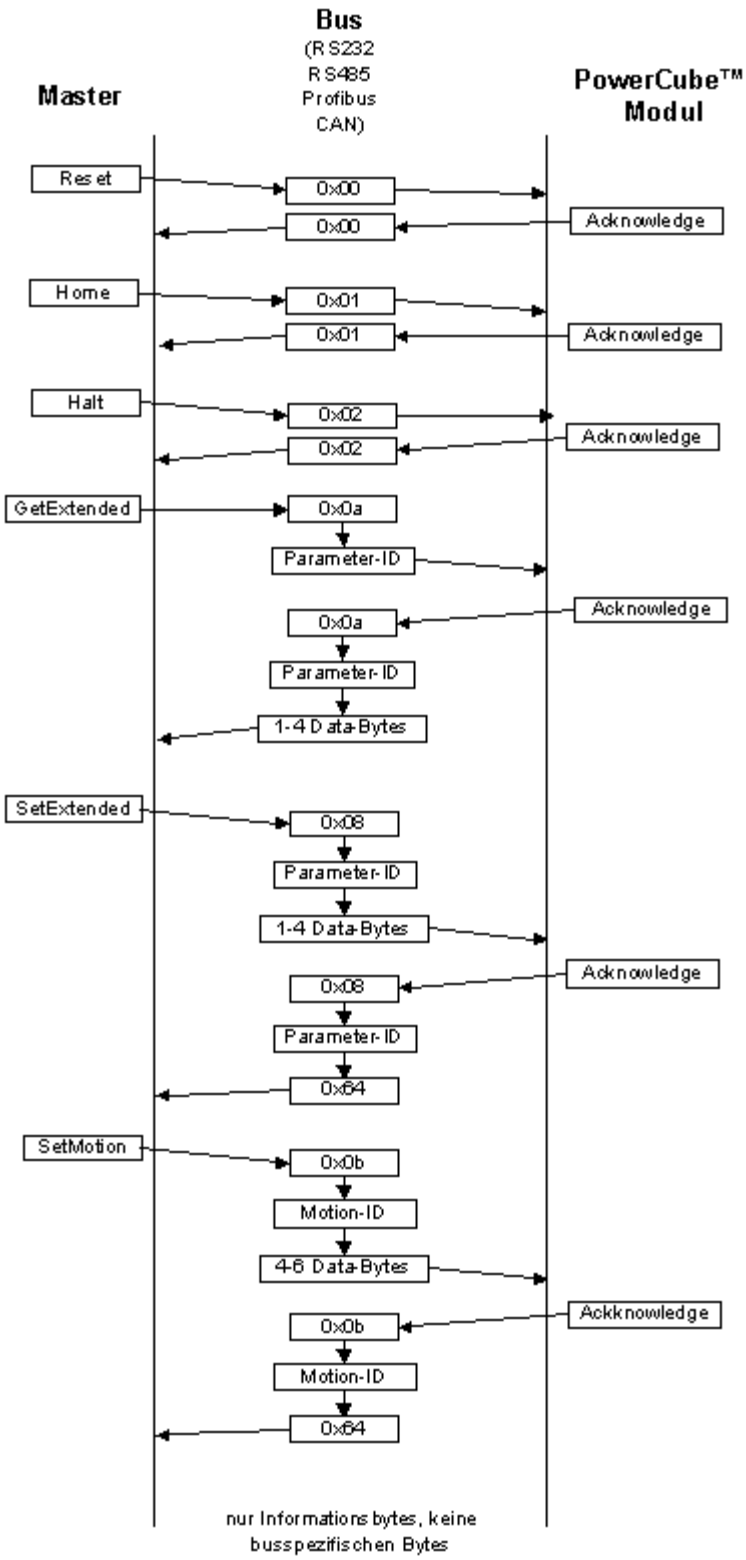
Für die Realisierung des Datenprotokolls ist die Kenntnis der im PowerCube™ verwendeten Datentypen erforderlich:

Datentyp	Bedeutung	Datenbreite	Wertebereich (dezimal)
Int8	Byte	8 Bit	-128 bis 127
UInt8	Byte, vorzeichenlos	8 Bit	0 bis 255
Int16	Wort	16 Bit	-32768 bis 32767
UInt16	Wort, vorzeichenlos	16 Bit	0 bis 65535
Int32	Doppelwort	32 Bit	-2147483848 bis 2147483847
UInt32	Doppelwort, vorzeichenlos	32 Bit	0 bis 4294967295
float	Gleitkommawert	32 Bit	$3.4\ e^{-038}$ bis $3.4\ e^{+038}$

Tabelle 35

7.8. Aufbau der Datenströme

Der Aufbau der Datenströme (Informationsbytes) zur Steuerung des PowerCube™ ist für alle Kommunikationsarten prinzipiell identisch. Im folgenden wird gezeigt, wie die einzelnen Kommandos aufgebaut werden. Das Diagramm zeigt nur Informationsbytes und keine busspezifischen Bytes.



7.9. Übersicht über die PowerCube™-Parameter ID's

Die folgende Tabelle enthält die Informationen über modulinterne Parameter, die über

die Kommunikation gelesen und geschrieben werden können.

Parameter	Para-meterID (Dec/Hex)	Bedeutung	Daten-typ	Read	Write
DefHomeOffset	0/0x00	Offset zur Referenzlage (Defaultwert). Mit Hilfe dieses Wertes kann ein Offset zur physikalischen Nullage (Referenzpunkt) des Antriebs eingestellt werden.	float	x	
DefGearRatio	1/0x01	Getriebeübersetzung (Defaultwert)	float	x	
DefLinRatio	2/0x02	Faktor zur Umrechung der Dreh- in Linearbewegung (Defaultwert)	float	x	
DefMinPos	3/0x03	Minimalposition des Antriebs (Defaultwert)	float	x	
DefMaxPos	4/0x04	Maximalposition des Antriebs (Defaultwert)	float	x	
DefMaxDeltaPos	5/0x05	Maximal zulässige Regelabweichung des Lagereglers: "Schleppabstand" (Defaultwert)	float	x	
DefMaxDeltaVel	6/0x06	Maximal zulässige Regelabweichung des Geschwindigkeitsreglers (Defaultwert)	float	x	
DefTorqueRatio	7/0x07	Faktor zur Umrechnung zwischen Stromaufnahme und ausgeübtem Moment (Defaultwert)	float	x	
DefCurRatio	8/0x08	Faktor zur Umrechnung der Rohdaten des Stromreglers in Ampere (Defaultwert)	float	x	
DefMinVel	9/0x09	Minimalgeschwindigkeit des Antriebs (Defaultwert)	float	x	
DefMaxVel	10/0x0a	Maximalgeschwindigkeit des Antriebs (Defaultwert)	float	x	
DefMinAcc	11/0x0b	Minimalbeschleunigung des Antriebs (Defaultwert)	float	x	
DefMaxAcc	12/0x0c	Maximalbeschleunigung des Antriebs (Defaultwert)	float	x	
DefMinCur	13/0x0d	Minimaler Sollwert für den Stromregler (Defaultwert)	float	x	
DefMaxCur	14/0x0e	Maximaler Sollwert für den Stromregler (Defaultwert)	float	x	
DefHomeVel	15/0x0f	Geschwindigkeit für die Referenzpunktfahrt. Dieser Wert ist vorzeichenbehaftet und legt auf diese Weise die Richtung fest. (Defaultwert)	float	x	
DefHomeAcc	16/0x10	Beschleunigung für die Referenzpunktfahrt (Defaultwert)	float	x	
DefCubeSerial	26/0x1a	Seriennummer des PowerCube™-Antriebs (Defaultwert)	UInt32	x	
DefConfig	27/0x1b	Konfigurationsdaten des Antriebs (Defaultwert)	UInt32	x	
DefPulsesPerTurn	28/0x1c	Anzahl der ausgewerteten Geberimpulse pro Motorumdrehung (Defaultwert)	UInt32	x	
DefCubeVersion	29/0x1d	Versionsinformation der Firmware (Defaultwert)	UInt16	x	
DefServiceInterval	30/0x1e	Wartungsintervall angegeben in Betriebsstunden (Defaultwert)	UInt16	x	

Tabelle 36

DefBrakeTimeOut	31/0x1f	Verzögerungszeit für das Zuschalten der Bremse nach Stillstand des Motors angegeben in ms (Defaultwert)	UInt16	x	
DefAddress	32/0x20	Busadresse des Moduls [1...31] (Defaultwert)	UInt8	x	
DefPrimBaud	34/0x22	Kennziffer für die voreingestellte Baudrate der CAN-Kommunkation (Defaultwert)	UInt8	x	
DefScndBaud	35/0x23	Kennziffer für die voreingestellte Baudrate der RS232/RS485-Kommunikation (Defaultwert)	UInt8	x	
PosCount	36/0x24	Absoluter Zählerstand des Positionsgebers (Istwert)	Int32	x	
RefPosCount	37/0x25	Zählerstand des Positionsgebers bei Erkennung des Referenzschalters (Istwert)	Int32	x	
DioSetup	38/0x26	Register der aktuell gesetzten Bits der internen DIOs (siehe PCube_getDioData)	UInt 32	x	x
CubeState	39/0x27	Statuswort des Antriebs (Istwert)	UInt32	x	
TargetPosInc	40/0x28	Sollwertvorgabe in Geberimpulsen für die Position (Zielwert)	UInt32	x	x
TargetVelInc	41/0x29	Sollwertvorgabe in Geberimpulsen/ms für die Geschwindigkeit (Zielwert)	UInt32	x	x
TargetAccInc	42/0x2a	Sollwertvorgabe in Geberimpulsen/ms^2 für die Beschleunigung (Ziel-wert)	UInt32	x	x
StepInc	43/0x2b	Schrittweite in Geberimpulsen des Antriebs im Stepbetrieb (Istwert)	UInt32	x	
HomeOffsetInc	44/0x2c	Nullpunktoffset des Antriebs in Geberimpulsen (Istwert)	Int32	x	
RawCur	53/0x35	Sollwert des digitalen Stromreglers in Digits [-500...+500] (Istwert/Vorgabe)	Int16	x	x
HomeToZeroInc	54/0x36	Anzahl der Geberimpulse zwischen Referenzschalter und Nullimpuls des Gebers (Istwert)	Int32	x	
Config	57/0x39	Konfigurationswort für den Antrieb (Vorgabe)	UInt32	x	x
MoveMode	58/0x3a	Betriebsart des Antriebs (Istwert)	UInt8	x	
IncRatio	59/0x3b	Verhältnis aus Geberimpulsen und Wegeinheit (Istwert)	float	x	
ActPos	60/0x3c	Aktuelle Position, vorberechnet in Wegeinheiten (Istwert)	float	x	
ActPos_	61/0x3d	Aktuelle Position im vergangenen Abtastzeitpunkt, vorberechnet in Wegeinheiten (Istwert)	float	x	
IPolPos	62/0x3e	Aktuelle Sollposition, vorberechnet in Wegeinheiten (Istwert)	float	x	
DeltaPos	63/0x3f	Differenz zwischen Soll- und Istposition, vorberechnet in Wegeinheiten = Regelabweichung (Istwert)	float	x	
MaxDeltaPos	64/0x40	Maximal zulässige Differenz zwischen Soll- und Istposition, vorberechnet in Wegeinheiten = max. zulässige Regelabweichung (Vorgabe)	float	x	x

Tabelle 36

ActVel	65/0x41	Aktuelle Geschwindigkeit des Antriebs, vorberechnet in Wegeinheiten/s (Istwert)	float	x	
IPolVel	66/0x42	Aktuelle Sollgeschwindigkeit, vorberechnet in Wegeinheiten/s (Istwert)	float	x	
MinPos	69/0x45	Endlage des Antriebs in Wegeinheiten = Minimalposition (Vorgabe)	float	x	x
MaxPos	70/0x46	Endlage des Antriebs in Wegeinheiten = Maximalposition (Vorgabe)	float	x	x
MaxVel	72/0x48	Maximal zulässige Geschwindigkeit des Antriebs in Wegeinheiten/s (Vorgabe)	float	x	x
MaxAcc	74/0x4a	Maximal zulässige Beschleunigung des Antriebs in Wegeinheiten/s^2; (Vorgabe)	float	x	x
MaxCur	76/0x4c	Maximal zulässiger Stromsollwert in Ampere für den digitalen Stromregler (Vorgabe)	float	x	x
Cur	77/0x4d	Aktueller Stromsollwert in Ampere des digitalen Stromreglers (Vorgabe)	float	x	x
TargetPos	78/0x4e	Sollwert für die Positionsvorgabe in Wegeinheiten (Zielwert)	float		x
TargetVel	79/0x4f	Sollwert für die Geschwindigkeitsvorgabe in Wegeinheiten/s (Zielwert)	float		x
TargetAcc	80/0x50	Sollwert für die Beschleunigungsvorgabe in Wegeinheiten/s^2; (Zielwert)	float		x
DefC0	81/0x51	Verstärkungsfaktor C0 (Defaultwert)	Int16	x	
DefDamp	82/0x52	Dämpfungsfaktor Damp (Defaultwert)	Int16	x	
DefA0	83/0x53	Nominalbeschleunigung des Motors (Defaultwert)	Int16		
ActC0	84/0x54	Verstärkungsfaktor C0 (Vorgabe)	Int16	x	x
ActDamp	85/0x85	Dämpfungsfaktor Damp (Vorgabe)	Int16	x	x
ActA0	86/0x86	Nominalbeschleunigung des Motors (Vorgabe)	Int16	x	x
DefBurnCount	87/0x87	Anzahl der bisher am Modul durchgeführten Brennvorgänge (Defaultwert)	UInt8	x	
Setup	88/0x88	Neu ab Version 3.5.00: 32-Bit-Wort, in dem u. a. Feedback-,DIO- und Kommunikationssetup hinterlegt sind (Default)	UInt32	x	
HomeOffset	89/0x89	Offset zur Referenzlage (Vorgabe)	float	x	x

Tabelle 36

Defaultwert: Diese Daten sind im ROM (Read Only Memory) des PowerCube™ gespeichert und werden nach dem Zuschalten der Betriebsspannung als Vorgabewerte aktiv.
Vorgabe: Diese Daten können für die Dauer des Betriebs vom Anwender verändert werden. Nach dem Abschalten der Betriebsspannung gehen die Einstellungen verloren und müssen nach dem erneuten Zuschalten wiederholt werden.
Zielwert: Dies sind die kurzzeitigen Zielvorgaben für Fahrkommandos. Sie beziehen sich ausschließlich auf Zielpositionen, -geschwindigkeiten und -beschleunigungen.
Die 32-Bit Konfigurationsworte Config und Setup (neu ab Modulversion 3.5.00) enthalten u.a. Einstellungen zur Bremsen-, Kommunikations-, Feedback- und Schalterkonfiguration. Setup kann zur Laufzeit nur gelesen, nicht jedoch verändert werden, während die Konfigurationsbits von Config zur Laufzeit neu gesetzt werden können. Nachfolgend ist die Bitbelegung ab Modulversion 3.5.00 wiedergegeben.

7.10. Konfigurationsbits im 32-Bit Wort Setup (Read Only)

Bit-ID (Hex)	Bedeutung
0x00000001	SETUP_ENCODER_FEEDBACK Lageinformation über Encoder Feedback
0x00000002	SETUP_RESOLVER_FEEDBACK Lageinformation über Resolver Feedback
0x00000004	SETUP_ABSOLUTE_FEEDBACK Lageinformation über Absolutgeber-Feedback
0x00000008	SETUP_4IN_4OUT Beschaltung des 15-poligen I/O-Interfaces Typ I
0x00000010	SETUP_3IN_ENCODER_IN Beschaltung des 15-poligen I/O-Interfaces TypII
0x00000020	SETUP_3IN_ENCODER_OUT Beschaltung des 15-poligen I/O-Interfaces TypIII
0x00000040	SETUP_RS232 Modulkommunikation über RS232-Schnittstelle
0x00000200	SETUP_CAN Modulkommunikation über CAN-Schnittstelle
0x00000400	SETUP_PROFIBUS Modulkommunikation über Profibus-Schnittstelle
0x00000800	SETUP_USE_M3ID CAN-Identifier für M3-Befehlssatz reservieren
0x00001000	SETUP_USE_M4ID CAN-Identifier für M4-Befehlssatz reservieren
0x00004000	SETUP_INVERT_MOTORDIR Umkehr der Motorlaufrichtung
0x00008000	SETUP_USE_SW2_AS_ENABLE High auf SW2-Datenleitung aktiviert Motorregelung
0x00010000	SETUP_USE_SW2_AS_BRAKE High auf SW2-Datenleitung löst Bremse
0x00020000	SETUP_ERROR_TO_OUT0 Im Fehlerfall sendet Modul ein Signal über Datenausgangskanal 0

Tabelle 37

7.11. Konfigurationsbits im 32-Bit Wort Config (ab Version 3.5.00)

Bit-ID (Hex)	Bedeutung
0x00000008	CONFIG_BRAKE_PRESENT Nach Abschluß der Bewegung wird Bremse automatisch aktiviert
0x00000010	CONFIG_BRAKE_AT_POWERON Nach Einschalten bleibt Bremse aktiv
0x00000020	CONFIG_SWR_WITH_ENCODERZERO Homing-Prozedur wird mit „Zero-Impuls“ des Encoders beendet
0x00000040	CONFIG_SWR_AT_FALLING_EDGE Referenzschalter arbeitet auf fallender Flanke
0x00000080	CONFIG_CHANGE_SWR_TO_LIMIT Nach Homing-Prozedur wird Referenzschalter zu Endlagenschalter
0x00000100	CONFIG_SWR_ENABLED Referenzschalter wird aktiviert

Tabelle 38

0x00000200	CONFIG_SWR_LOW_ACTIVE Referenzschalter ist low-aktiv
0x00000400	CONFIG_SWR_USE_EXTERNAL Der externe Referenzschalter-Eingang wird ausgewertet
0x00000800	CONFIG_SW1_ENABLED Endlagenschalter 1 wird aktiviert
0x00001000	CONFIG_SW1_LOW_ACTIVE Endlagenschalter 1 low-aktiv
0x00002000	CONFIG_SW1_USE_EXTERNAL Externer Endlagenschalter-Eingang 1 wird ausgewertet
0x00004000	CONFIG_SW2_ENABLED Endlagenschalter 2 wird aktiviert
0x00008000	CONFIG_SW2_LOW_ACTIVE Endlagenschalter 2 low-aktiv
0x00010000	CONFIG_SW2_USE_EXTERNAL Externer Endlagenschalter-Eingang 2 wird ausgewertet
0x00020000	CONFIG_LINEAR_DRIVE Modultyp ist Linearmodul
0x00100000	CONFIG_M3_COMPATIBLE Befehlssatz und Verhalten des Moduls ist kompatibel zu MORSE3
0x00200000	CONFIG_LINEAR_SCREW Modultyp ist Linearmodul mit Spindeltrieb
0x00400000	CONFIG_NOTIFYEMPTYBUFFER Benachrichtigung bei leerem Befehlspuffer im Step-Bewegungsmodus
0x00800000	CONFIG_DISABLE_ON_HALT Bei Setzen des HALT-Flags wird Motorregelung abgeschaltet
0x01000000	CONFIG_WATCHDOG_ENABLE Watchdog-Funktion im Modul wird freigeschaltet
0x02000000	CONFIG_ZERO_MOVE_AFTER_HOK Modul wird nach Referenzfahrt auf benutzerdefinierte Nullposition bewegt
0x04000000	CONFIG_DISABLE_ACK Keine Bestätigung vom Modul nach SET-Kommando

Tabelle 38

7.12. Konfigurationsbits im 32-Bit Wort Config (bis Version 2.5.1x)

Bit-ID (Hex)	Bedeutung
0x00000001	CONFIG_ENCODER_FEEDBACK Lageinformation über Encoder Feedback
0x00000002	CONFIG_RESOLVER_FEEDBACK Lageinformation über Resolver Feedback
0x00000004	CONFIG_ABSOLUTE_FEEDBACK Lageinformation über Absolutgeber-Feedback
0x00000008	CONFIG_BRAKE_PRESENT Nach Abschluß der Bewegung wird Bremse automatisch aktiviert
0x00000010	CONFIG_BRAKE_AT_POWERON Nach Einschalten bleibt Bremse aktiv
0x00000020	CONFIG_SWR_WITH_ENCODERZERO Homing-Prozedur wird mit „Zero-Impuls“ des Encoders beendet
0x00000040	CONFIG_SWR_AT_FALLING_EDGE Referenzschalter arbeitet auf fallender Flanke
0x00000080	CONFIG_CHANGE_SWR_TO_LIMIT Nach Homing-Prozedur wird Referenzschalter zu Endlagenschalter
0x00000100	CONFIG_SWR_ENABLED Referenzschalter wird aktiviert
0x00000200	CONFIG_SWR_LOW_ACTIVE Referenzschalter ist low-aktiv
0x00000400	CONFIG_SWR_EQ_POSLIMIT SWR fungiert als Endlagenschalter unfd markiert die Begrenzung in positive Bewegungsrichtung
0x00000800	CONFIG_SW1_ENABLED Endlagenschalter 1 wird aktiviert
0x00001000	CONFIG_SW1_LOW_ACTIVE Endlagenschalter 1 low-aktiv
0x00002000	CONFIG_SW1_EQ_POSLIMIT Markiert die Begrenzung in positive Bewegungsrichtung
0x00004000	CONFIG_SW2_ENABLED Endlagenschalter 2 wird aktiviert
0x00008000	CONFIG_SW2_LOW_ACTIVE Endlagenschalter 2 low-aktiv
0x00010000	CONFIG_SW2_EQ_POSLIMIT Markiert die Begrenzung in positive Bewegungsrichtung
0x00020000	CONFIG_LINEAR_DRIVE Modultyp ist Linearmodul
0x00040000	CONFIG_ADE_PRESENT #) Analog/Digital-Erweiterung im Modul installiert
0x00040000	CONFIG_OPENBRAKE_ON_SW2 *) High auf SW2-Datenleitung löst Bremse
0x00080000	CONFIG_PID_OUT_TO_DAC Datenausgabe der Regelschleife wird Digital/Analog-Konverter übergeben
0x00100000	CONFIG_M3_COMPATIBLE Befehlssatz und Verhalten des Moduls ist kompatibel zu MORSE3
0x00200000	CONFIG_LINEAR_SCREW Modultyp ist Linearmodul mit Spindeltrieb

Tabelle 39

0x00400000	CONFIG_NOTIFYEMPTYBUFFER +) Benachrichtigung bei leerem Befehlspuffer im Step-Bewegungsmodus
0x40000000	CONFIG_CAN_RS232_COMM Modulkommunikation über CAN- bzw. RS232-Schnittstelle
0x80000000	CONFIG_PROFI_RS485_COMM Kommunikation über Profibus-Schnittstelle

Tabelle 39

#) gültig bis einschließlich Betriebssystemversion 2.5.04
*) gültig ab Betriebssystemversion 2.5.10
+) gültig ab Betriebssystemversion 2.5.03

CAN

	CAN-ID	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send*	0x0F1	0x01 Home	-	-	-	-	-	-	-
Ack**	0x0B1	0x01	-	-	-	-	-	-	-
Send*	0x0F1	0x00 Reset	-	-	-	-	-	-	-
Ack**	0x0B1	0x00	-	-	-	-	-	-	-
Send*	0x0F1	0x08 SetExtended	0x50 TargetAcc	0x00	0x00	0x00	0x3f	-	-
Ack**	0x0B1	0x08	0x50	0x64	-	-	-	-	-
Send*	0x0F1	0x08 SetExtended	0x4f TargetVel	0x0a	0xd7	0xa3	0x3d	-	-
Ack**	0x0B1	0x08	0x4f	0x64	-	-	-	-	-
Send*	0x0F1	0x0b SetMotion	0x04 FRAMP_MODE	0x00	0x00	0x40	0x3f	-	-
Ack**	0x0B1	0x0b	0x04	0x64	-	-	-	-	-
Send***	0x0D1	0x0a GetExtended	0x27 CubeState	-	-	-	-	-	-
Ack**	0x0B1	0x0a	0x27	State	State	State	State	-	-
Send***	0x0D1	0x0a GetExtended	0x3c ActPos	-	-	-	-	-	-
Ack**	0x0B1	0x0a	0x3c	Pos	Pos	Pos	Pos	-	-

Tabelle 40

CAN-ID = CANID_CMDPUT + CubeID = 0x0e0 + 0x011 = 0x0F1
***CAN-ID = CANID_CMDGET + CubeID = 0x0c0 + 0x011 = 0x0D1
***CAN-ID = CANID_CMDGET + CubeID = 0x0c0 + 0x011 = 0x0D1

CAN

	CAN-ID	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send*	0x0F1	0x01 Home	-	-	-	-	-	-	-
Ack**	0x0B1	0x01	-	-	-	-	-	-	-

Tabelle 41

Send*	0x0F1	0x00 Reset	-	-	-	-	-	-	-
Ack**	0x0B1	0x00	-	-	-	-	-	-	-
Send*	0x0F1	0x08 SetExtended	0x50 TargetAcc	0x00	0x00	0x00	0x3f	-	-
Ack**	0x0B1	0x08	0x50	0x64	-	-	-	-	-
Send*	0x0F1	0x08 SetExtended	0x4f TargetVel	0x0a	0xd7	0xa3	0x3d	-	-
Ack**	0x0B1	0x08	0x4f	0x64	-	-	-	-	-
Send*	0x0F1	0x0b SetMotion	0x04 FRAMP_MODE	0x00	0x00	0x40	0x3f	-	-
Ack**	0x0B1	0x0b	0x04	0x64	-	-	-	-	-
Send***	0x0D1	0x0a GetExtended	0x27 CubeState	-	-	-	-	-	-
Ack**	0x0B1	0x0a	0x27	State	State	State	State	-	-
Send***	0x0D1	0x0a GetExtended	0x3c ActPos	-	-	-	-	-	-
Ack**	0x0B1	0x0a	0x3c	Pos	Pos	Pos	Pos	-	-

Tabelle 41

Ab PowerCube™-Betriebssystemversion 2.5.16 bzw. 3.5.13 ist es möglich, FRAMP_ACK anstelle von FRAMP_MODE zu verwenden. In diesem Fall enthält das Acknowledge des Moduls Statusinformationen wie aktuelle Position, Kurzstatus und DIO-Status in komprimierter Form. Die 8 Bit des Kurzstatus sind wie folgt definiert::

Bit-Zuordnung	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Bedeutung	FULL	INPROG	END	MOTION	SW2	SW1	SWR	NOK *
Entspricht dem CubeState Bit	FULL-BUFFER	INPRO-GRESS	RAMP_END	MOTION	SW2	SW1	SWR	-

Tabelle 42

* NOK (NOT_OK) setzt sich aus einer logischen ODER-Verknüpfung der Flags STATE_ERROR, !STATE_HOK und STATE_HALTED zusammen.
Die letzten beiden Zeilen des obigen Beispiels würden dann also wie folgt aussehen::

Send	0x0f1	0x0b SetMotion	0x04 FRAMP_ACK	0x00 Pos	0x00 Pos	0x40 Pos	0x3f Pos	-	-
Ack	0x0f1	0x0b	0x0e	Act-Pos	Act-Pos	Act-Pos	Act-Pos	Short State	DIO

Tabelle 43

RS232

	STX	TEL-ID	TEL-ID	Cmd ID	Par-ID	B2	B3	B4	B5	B6	B7	BCC	ETX
Send	0x02	0x06	0x21	0x01	-	-	-	-	-	-	-	0x28	0x03
Ack	0x02	0x0a	0x21	0x01	-	-	-	-	-	-	-	0x2c	0x03
Send	0x02	0x06	0x21	0x00	-	-	-	-	-	-	-	0x27	0x03
Ack	0x02	0x0a	0x21	0x00	-	-	-	-	-	-	-	0x2b	0x03

Tabelle 44

Send	0x02	0x06	0x26	0x08	0x50	0x00	0x00	0x00	0x3f	-	-	0xc3	0x03
Ack	0x02	0x0a	0x26	0x08	0x50	0x64	-	-	-	-	-	0xec	0x03
Send	0x02	0x06	0x26	0x08	0x4f	0x0a	0xd7	0xa3	0x3d	-	-	0x46	0x03
Ack	0x02	0x0a	0x26	0x08	0x4f	0x64	-	-	-	-	-	0xeb	0x03
Send	0x02	0x06	0x26	0x0b	0x04	0x00	0x00	0x40	0x3f	-	-	0xba	0x03
Ack	0x02	0x0a	0x26	0x0b	0x04	064	-	-	-	-	-	0xa3	0x03

Tabelle 44

Die Berechnung der 2 TELID-Bytes möge folgendes Beispiel (SetExtended) verdeutlichen:

	1. Byte								2. Byte							
Bit-Zuordnung	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TELID_SENDDAT	0	0	0	0	0	1	m	m	m	m	m	0	n	n	n	n
TELID_RECVDAT	0	0	0	0	1	0	m	m	m	m	m	0	n	n	n	n

Tabelle 45

Beispiel Senden:

TELID_SENDDAT Modul ID = 17 Command SetExtended	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0
	TELID Byte1 = 0x06								TELID Byte2 = 0x26							

Tabelle 46

Beispiel Empfangen:

TELID_RECVDAT Modul ID = 17 Command SetExtended	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1	0
	TELID Byte1 = 0x0a								TELID Byte2 = 0x26							

Tabelle 47

Anhand des Kommandos Halt (ID: 0x02) soll die DLE-Korrektur veranschaulicht werden:

Senden ohne Korrektur:

```
STX  TELID  TELID  CMD   BCC   ETX
0x02 0x06   0x21   0x02  0x29  0x03
```

Senden mit Korrektur:

```
STX  TELID  TELID  CMD (m.DLE) BCC  ETX
0x02 0x06   0x21   0x10  0x82  0x29 0x03
```

Profibus

	Byte 0 ComandID	Byte 1 ParamID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x01 Home	-	-	-	-	-	-	X
Ack	0x01	-	-	-	-	-	-	-
Send	0x00 Reset	-	-	-	-	-	-	X
Ack	0x00	-	-	-	-	-	-	-
Send	0x08 SetExtended	0x50 TargetAcc	float Acc	float Acc	float Acc	float Acc	-	X
Ack	0x08	0x50	0x64	-	-	-	-	-

Send	0x08 SetExtended	0x4f TargetVel	float Vel	float Vel	float Vel	float Vel	-	X
Ack	0x08	0x4f	0x64	-	-	-	-	-
Send	0x0b SetMotion	0x04 FRAMPMODE	float Pos	float Pos	float Pos	float Pos	-	X
Ack	0x0b	0x04	064	-	-	-	-	-

Tabelle 47

X: Bei der Übertragung von Daten über den Profibus ist darauf zu achten, daß im Byte 7 ein veränderlicher Wert, z. B. ein inkrementeller Counter gesetzt wird. Denn beim Senden identischer Telegramme wird nur das erste ausgewertet (beispielsweise mehrere Bewegungsrampen mit gleichen Parametern) !

7.13. Kommandos an alle verbundenen Module in einer Kommandozeile senden

Mit Hilfe des schon erwähnten CAN-Identifiers CANID_CMDALL ist es möglich, in einer einzigen Kommandozeile folgende Befehle an alle verbundenen Module zu schicken. Diese Befehle werden besonders schnell verarbeitet; es gibt allerdings von Seiten der Module kein Acknowledge.

CANID_CMDALL (0x100)	Byte 0 ComandID	Byte1 Param.ID	Bedeutung
	0x00	-	Reset
	0x02	-	Halt
	0x07	-	Watchdog-Refresh
	0x09	0x00 ... 0x03 (*)	Change Baudrate

Tabelle 48

(*) Baudrate: 0 = 75K, 1 = 250K, 2 = 500K, 3 = 1000K.
Eine ausführliche Erläuterung der Befehle ist im Abschnitt "Die API-Schnittstelle zur PowerCube™-Steuerung" vorhanden.

7.14. Betriebsverhalten des PowerCube™

Die umfangreichen Funktionen des PowerCube™-Betriebssystems bedingen für den störungsfreien Betrieb des Antriebs einige wichtige Regeln, die unbedingt Beachtung finden sollten. Dies bezieht sich auf den Ablauf bei der Vorgabe von Bewegungssequenzen und Maßnahmen zur Überwachung des Systems. Die Auslegung der Mastersteuerung ist meist abhängig vom konkreten Anwendungsfall. Bei der Auslegung der Steuerung sollten folgende Faktoren betrachtet werden:

- Echtzeitverhalten der Anlage und zeitliche Anforderungen für den Betrieb
- Verkopplung von mehreren Antrieben in einer kinematischen Kette
- Sicherheitsanforderung beim Betrieb der Gesamtanlage.

In jedem Fall kann die Nutzung des PowerCube™ für die besonderen Anforderungen der Anwendung optimiert werden. Dazu erhalten Sie in den folgenden Abschnitten nähere Informationen.

7.15. Betriebsweise im Normalfall

Nach Zuschalten der Betriebsspannung ist der PowerCube™ sofort betriebsbereit. Die

rückgemeldete Position entspricht dem voreingestellten Homeoffset. Der Antrieb kann mit den Bewegungs-kommandos verfahren werden.
Bei den Modulen handelt es sich um mechatronische Elemente, deren Bauteile fettgeschmiert sind. *Die Maximaldaten der Module sollten also erst nach einer Warmlaufphase abgefordert werden*.
Üblicherweise ist die erste Bewegung des Antriebs nach dem Zuschalten der Betriebsspannung die Referenzpunktfahrt (Homing). Dies ist notwendig, weil die Einstellungen des Antriebs hinsichtlich Maximal- und Minimalposition auf die Nullage des Antriebs bezogen sind (Nullage ist Referenzpunkt abzüglich Homeoffset). Die Begrenzung der Lagesollwerte auf die voreingestellten Daten funktioniert demzufolge nur nach erfolgreicher Referenzpunktfahrt. Sie sollten aus diesem Grund folgende Hinweise berücksichtigen:
Vorgabe von Bewegungskommandos erst nach erfolgreicher Referenzpunktfahrt,

- Ist eine Referenzpunktfahrt nicht möglich, so nur langsame Bewegungen auszulösen , die kontrolliert und im Notfall abgeschaltet werden können,
- Der Antrieb ist vor dem Abschalten der Betriebsspannung stets in die Referenzlage zurückzufahren.
- Im Statuswort des Antriebs wird durch das Bit STATE_HOME_OK signalisiert, daß eine Referenzfahrt erfolgreich war.

Die Vorgabe von Daten an den Antrieb wird vom Betriebssystem des PowerCube™ überwacht. Im einzelnen bedeutet dies:
Überwachung der Vorgaben von Zielposition, Geschwindigkeit, Beschleunigung und Strom bei Fahrkommandos. Die Vorgaben werden auf die voreingestellten Maximalwerte begrenzt. Vorgaben kleiner als die voreingestellten Minimalwerte werden ebenfalls begrenzt (siehe unter "Lesen und Schreiben von Parametern": Zielwerte).
Überwachung der Vorgaben beim Setzen von Parametern. Dies bezieht sich auf alle Werte, die über die Kommunikation für die Dauer des Betriebs verändert werden können (siehe unter "Lesen und Schreiben von Parametern": Vorgabe).

7.16. Die Benutzung der Modul-Watchdogfunktion

Für jedes angeschlossene Modul muß sichergestellt sein, daß im Konfigurationswort Config der Watchdog verfügbar gemacht worden ist (Flag CONFIG_WATCHDOG_ENABLE).
An jedes angeschlossene Modul muß die Kommandosequenz CANID_CMDPUT (0x0e0 + Modul-ID) + CommandID 0x07 gesendet werden, um den modulinternen Watchdog online zu schalten. Ein erneutes Senden dieser Sequenz schaltet den Watchdog zurück in den offline-Zustand.
In einem Intervall kleiner 50 ms muß die Kommandosequenz CANID_CMDALL (0x100) + CommandID 0x07 gesendet werden, die bei allen angeschlossenen Modulen das Watchdog-Refresh auslöst.
Falls das Watchdog-Refresh nicht zyklisch innerhalb dieser Zeitspanne ausgelöst wird, werden alle Module in einen Haltezustand versetzt, wobei sie die Statusflags STATE_COMM_ERROR und STATE_HALTED anzeigen.
Um die Module wieder zu aktivieren, kann man nun entweder den Watchdog jedes einzelnen Moduls offline schalten. Anschließend sind die Module mit einem RESET zurücksetzen. Es kann aber auch erneut zyklisch ein Watchdog-Refresh und mittels der Kommandosequenz CANID_CMDALL (0x100) + CommandID 0x00 ein RESET gleichzeitig an alle Module gesendet werden

7.17. Der Modulstatus: Das Statuswort CubeState

Der Zustand des Antriebs kann jederzeit mit Hilfe des Statusworts "CubeState" ermittelt werden. Anhand des Modulstatus erkennt der Anwender den augenblicklichen Ausführungsszustand von Befehlen, kann Fehlermeldungen und Ursachen für Fehler ermitteln. Es ist in jedem Fall ratsam, das Statuswort in regelmäßigen Abständen zu

pollen. Die Häufigkeit wird hierbei von der Anwendung bestimmt. Das Statuswort ist ein vorzeichenloser 32-Bit Integer-Wert, dessen Information bitweise organisiert ist. Jedem Bit ist eine Statusinformation zugeordnet. Im Modulstatus können mehrere Bits gleichzeitig gesetzt sein. Dies ist der Aufbau des CubeState:

Flag	Bit	Wert	Bedeutung
STATE_ERROR	0	0x00000001	Es liegt ein Fehler im Modul vor. Der Antrieb löst einen Nothalt aus und nimmt keine weiteren Bewegungskommandos an. Die Fehlerursache kann anhand der Fehlerbits ermittelt werden. In vielen Fällen kann der Anwender einen Fehlerzustand durch ein RESET-Kommando quittieren und mit der Arbeit fortfahren.
STATE_HOME_OK	1	0x00000002	Dieses Bit wird nach erfolgreicher Referenzpunktfahrt gesetzt. Es bedeutet, daß der Antrieb den Nullpunkt in seinem Koordinatensystem eindeutig zugeordnet hat und alle voreingestellten Werte, die die Positionsbegrenzung betreffen Gültigkeit haben. Dieses Bit wird bei Beginn jeder Referenzpunktfahrt (Kommando HOME) gelöscht.
STATE_HALTED	2	0x00000004	Dieses Bit wird im Zusammenhang mit einem Nothalt gesetzt. Es besagt, daß der Antrieb steht und sich in einem Sicherheitszustand befindet. Bevor der Antrieb wieder Bewegungskommandos ausführt, muß er ein RESET-Kommando erhalten haben und eventuelle Fehlerbits gelöscht haben. Nothalt kann durch die interne Fehlerüberwachung oder durch das HALT-Kommando ausgelöst werden.
STATE_POWERFAULT	3	0x00000008	Dieses Bit gibt einen Fehler der PowerCube™-Endstufe an. In den meisten Fällen wird nach Auftreten dieses Fehlers ein Abschalten der Betriebsspannung notwendig sein. Der Fehler wird genauer durch die Bits 18 bis 23 beschrieben.
STATE_TOW_ERROR	4	0x00000010	Schleppfehler. Der Lageregler konnte der Sollwertvorgabe nicht folgen und hat eine nicht zulässige Regelabweichung detektiert. Die maximal zulässige Regelabweichung kann anhand des Wertes „MaxDeltaPos" eingestellt werden.
STATE_COMM_ERROR	5	0x00000020	Ein Fehler bei der Datenübertragung ist aufgetreten. Sollte dieser Fehler häufig auftreten, ist eine Untersuchung der Verbindungskabel sinnvoll. Prüfen Sie, ob der Bus mit einem Widerstand abgeschlossen wurde.
STATE_SWR	6	0x00000040	Dieses Bit zeigt den Zustand des Referenzschalters an. Ist es gesetzt, so ist der Referenzschalter aktiv (positive Logik).
STATE_SW1	7	0x00000080	Dieses Bit zeigt den Zustand des Endlagenschalters "1" an. Ist es gesetzt, so ist der Endlagenschalter "1" aktiv (positive Logik), die Endlage wurde überschritten.
STATE_SW2	8	0x00000100	Dieses Bit zeigt den Zustand des Endlagenschalters "2" an. Ist es gesetzt, so ist der Endlagenschalter "2" aktiv (positive Logik), die Endlage wurde überschritten.
STATE_BRAKEACTIVE	9	0x00000200	Dieses Bit zeigt den Zustand der Bremse an. Es wird nur verwendet, wenn eine Bremse konfiguriert wurde. Ist das Bit gesetzt, so ist die Bremse aktiviert und übt ihre Wirkung aus.
STATE_CURLIMIT	10	0x00000400	Dieses Bit ist eine Warnung des Lagereglers, dessen Stellwertbegrenzung aktiviert wurde. Der Antrieb befindet sich mit den aktuellen Einstellungen im Grenzlastbereich und dem Regler fehlt die notwendige Regelreserve. Das Bit kann durch RESET gelöscht werden.
STATE_MOTION	11	0x00000800	Dieses Bit zeigt an, daß sich der Antrieb in Bewegung befindet. Nach Stillstand wird es automatisch gelöscht.
STATE_RAMP_ACC	12	0x00001000	Dieses Bit zeigt an, daß der Antrieb in die Beschleunigungsphase beim Rampenbetrieb eingetreten ist. Es wird erst nach Beendigung der Rampenfahrt gelöscht.
STATE_RAMP_STEADY	13	0x00002000	Dieses Bit zeigt an, daß der Antrieb in die Phase der gleichförmigen Bewegung beim Rampenbetrieb eingetreten ist. Es wird erst nach Beendigung der Rampenfahrt gelöscht.

Tabelle 49

STATE_RAMP_DEC	14	0x00004000	Dieses Bit zeigt an, daß der Antrieb in die Verzögerungs-phase beim Rampenbetrieb eingetreten ist. Es wird erst nach Beendigung der Rampenfahrt gelöscht.
STATE_RAMP_END	15	0x00008000	Mit diesem Bit signalisiert der Antrieb, daß die Rampenfahrt beendet wurde und sich der Motor im Stillstand befindet.
STATE_INPROGRESS	16	0x00010000	Dieses Bit hat nur im Stepbetrieb Bedeutung. Ein STEP-Kommando befindet sich in Ausführung.
STATE_FULLBUFFER	17	0x00020000	Dieses Bit hat nur im Stepbetrieb Bedeutung. Es wird gesetzt, wenn STATE_INPROGRESS gesetzt ist und ein weiteres STEP-Kommando empfangen wurde. Das zuletzt empfangene STEP-Kommando wird in den Puffer übernommen und nach Ausführung des vorhergehenden STEP-Kommandos automatisch "angehängt".
STATE_POW_VOLT_ERR	18	0x00040000	Die Zwischenkreisspannung des Motors ist unter den zulässigen Minimalwert gesunken. Die Endstufe wird abgeschaltet. Dieser Fehler kann durch RESET zurückgesetzt werden.
STATE_POW_FET_TEMP	19	0x00080000	Die Temperatur der Endstufentransistoren hat das zulässige Maximum überschritten. Die Endstufe wird abgeschaltet. Die Betriebsspannung muß abgeschaltet werden.
STATE_POW_WDG_TEMP	20	0x00100000	Die Temperatur der Motorwicklungen hat das zulässige Maximum überschritten. Die Endstufe wird abgeschaltet. Die Betriebsspannung muß abgeschaltet werden.
STATE_POW_SHORTCUR	21	0x00200000	Es ist ein Kurzschluß in der Endstufe aufgetreten. Die Endstufe wird abgeschaltet. Die Betriebsspannung muß abgeschaltet werden. Konsultieren Sie Ihren PowerCube™-Servicepartner.
STATE_POW_HALLERR	22	0x00400000	Es ist ein Fehler bei der Erfassung der Hallimpulse aufgetreten. Die Endstufe wird abgeschaltet. Die Betriebsspannung muß abgeschaltet werden.
STATE_POW_INTEGRAL-ERR	23	0x00800000	Die Dauerbelastung des Motors hat die Grenze der Zulässigkeit überschritten. Die Endstufe wird abgeschaltet. Die Betriebsspannung muß abgeschaltet werden. Prüfen Sie die Belastung des Antriebs und die Vorgaben für Geschwindigkeit und Beschleunigung.
STATE_CPU_OVERLOAD	24	0x01000000	Die CPU ist überlastet. Die Endstufe wird abgeschaltet. Die Betriebsspannung muß abgeschaltet werden. Wenden Sie sich an Ihren PowerCube™-Servicepartner.
STATE_POW_TIMEOUT *	24	0x01000000	Kommunikation zwischen CPU und Stromregler ist unterbrochen. Die Betriebsspannung muß abgeschaltet werden. Wenden Sie sich an Ihren PowerCube™-Ser-vicepartner. *Ab Modulversion 3.5.14 verfügbar. Ersetzt STATE_CPU_OVERLOAD.
STATE_BEYOND_HARD	25	0x02000000	Der PowerCube™ ist außerhalb des zulässigen Bewegungsbereichs. Um den Antrieb mit Motorkraft wieder aus der Endlage zu befreien, ist eine vorgeschriebene Prozedur einzuhalten. Siehe dazu "Betriebsweise bei Störungen".
STATE_BEYOND_SOFT	26	0x04000000	Der PowerCube™ hat die Software-Endlagen überschritten. Es wird ein Nothalt ausgeführt. Dieser Fehler kann durch RESET gelöscht werden.
STATE_POW_SETUP_ERR *	27	0x08000000	Fehler bei der Initialisierung des Stromreglers, Mo-duleinstellung stimmt nicht mit Reglerkonfiguration überein (5A/10A Typen). Die Betriebsspannung muß abgeschaltet werden. Wenden Sie sich an Ihren PowerCube™-Servicepartner. *Ab Modulversion 3.5.14 verfügbar.

Tabelle 49

7.18. Betriebsweise bei Störungen

Die Vorgabe von Daten an den Antrieb wird vom Betriebssystem des PowerCube™

überwacht. Im einzelnen bedeutet dies:
Fehler, die während des Betriebes auftreten, werden Ihnen im Statusworts "CubeState" mitgeteilt. Bei den Fehlermeldungen sollte man zwischen den zurücksetzbaren Fehlern und den Fehlern unterscheiden, die ein Abschalten des Moduls erfordern .
Folgende Fehler erfordern ein sofortiges Abschalten des Moduls:
Temperaturfehler in der Endstufe oder in der Motorwicklung. Wenn Sie Ihre Module im Dauerbetrieb überlasten, kann es zu diesem Fehler kommen. Bitte überdenken Sie den Einsatz der Module.
Kurzschluß in der Endstufe, Hallgeberfehler. Diese Fehler können eigentlich nur durch äußere Gewalteinwirkung auftreten. Bitte wenden Sie sich an Ihren Service-Partner.
Die folgenden Fehler können zurückgesetzt werden:

- Schleppfehler
- Fehler durch Überfahren der Endlagenschalter.
- Fehler in der Datenübertragung
- Zusammenbruch der Zwischenkreisspannung

8. PC-Programmierschnittstellen

Die von der amtec GmbH entwickelten intelligenten Antriebsmodule des PowerCube™-Systems werden über eine serielle Kommunikationsschnittstelle angesteuert. Das Prinzip der Kommunikation mit den Antriebsmodulen und die dafür verfügbare Systemsoftware werden in dem vorliegenden Programmierhandbuch beschrieben.
In diesem Programmierhandbuch sind alle verfügbaren Funktionsbibliotheken des PowerCube™-Systems beschrieben. Bitte beachten Sie, daß nur die Abschnitte für Ihre Anwendung Gültigkeit haben, für die Sie die Funktionsbibliotheken erworben haben.
Die für alle Bibliotheken verwendete, einheitliche *DriveLib* -Programmierschnittstelle stellt eine Treiber-Schnittstelle zur Ansteuerung von intelligenten Positionierantrieben des PowerCube™-Systems dar. Auf der Basis dieser Programmierschnittstelle entwickelte Applikationen erlauben den Einsatz aller Funktionsbibliotheken des PowerCube™-Systems.

API

Application Program Interface.
Diese Programmierschnittstelle (im folgenden API genannt) ist von der konkreten Implementierung unabhängig. Das heißt , unterschiedliche Treibervarianten werden über die gleiche Programmierschnittstelle angesteuert.

Leistungsumfang

Im einzelnen bietet das API mit seinen einfach zu handhabenden Funktionen folgende Leistungen:
Ansteuerung der PowerCube™-Antriebsmodule (m5dll)
Die konkreten Treiber des API stehen Ihnen in Form vorcompilierter dynamischer Funktionsbibliotheken (DLL) zur Einbindung in Ihre Anwendersoftware zur Verfügung.

8.1. Systemvoraussetzungen

Überzeugen Sie sich bitte zunächst davon, daß Ihr Rechnersystem die nötigen Voraussetzungen zur Anwendung der PowerCube™-Systemsoftware erfüllt.
Der verwendete PC muß ein IBM-kompatibler PC mit mindestens einem Pentium-133 Prozessor sein.
Es muß Windows 95, 98 oder NT installiert sein.
Die gegebenenfalls notwendige Hardware-Schnittstelle muß installiert sein und der Treiber dieser Hardwareschnittstelle (z. B. für CAN) muß geladen sein.

8.2. Allgemeine Hinweise zur Installation

Die für die Anwendung der einzelnen Funktionsbibliotheken notwendigen Dateien (C-Headerdateien, LIB-Dateien) sind in einem Verzeichnis zusammengefaßt. Im Lieferumfang enthalten ist generell auch ein einfaches Testprogramm, das Ihnen den Einstieg in die Anwendung der PowerCube™-Systemsoftware erleichtern soll.
Die Kommunikation der Softwareschnittstelle zu den Modulen erfordert es , daß entsprechende Treiber im System installiert wurden, um eine Kommunikation zwischen m5dll und dem PowerCube™-Modul herzustellen. Diese Treiber sind nicht Bestandteil der m5dll und deren Installation ist herstellerspezifisch und muß gegebenenfalls im Handbuch des Hardwareanbieters nachgelesen werden.
Von der aktuell vorliegenden Version der m5dll werden folgende Schnittstellen unterstützt:

- CAN Interfaces der Firma IXXAT
- RS232 Standard RS232-Schnittstelle des Rechners.
- NET Kommunikation über TCP/IP-Netze

Die Kommunikation über die RS232-Schnittstelle erfolgt ohne die Notwendigkeit einer Treiberinstallation, da diese bereits im Betriebssystem enthalten sind.
Zur Kommunikation über die VCI (CAN) Schnittstelle ist es erforderlich, die VCI-Treiber

der Firma IXXAT zu installieren.
Sollte die Netzwerkschnittstelle verwendet werden, so muß auf dem Clientrechner die WINSOCK2 Schnittstelle im Betriebssystem installiert sein. Dies ist unter Windows NT 4.0 SP3 und Windows 98 standardmäßig der Fall. Des weiteren wird das Netzwerkprotokoll TCP/IP benötigt, das auch korrekt installiert sein muß.

8.3. Typographie

Zur Hervorhebung und übersichtlicheren Darstellung bestimmter Sachverhalte werden innerhalb dieses Handbuchs verschiedene Schriftarten verwendet:
reserved: reservierte Wörter, Namen von Windows-API-Funktionen etc.
`listing`: Programmlistings von Beispielprogrammen, Namen von API-Funktionen, Elemente der Programmiersprache
parameter Parameter und Rückgabewerte der API-Funktionen.

8.4. Grundlagen der Programmierung mit PowerCube™-Systembibliotheken

Die PowerCube™-Systembibliotheken lassen sich - unabhängig von der unterstützten Thematik - nach einem bestimmten Schema anwenden. Wie bereits eingangs geschildert, ist die Schnittstellendefinition unabhängig von der konkreten Implementierung. Darüberhinaus sind einige allgemeine Funktionen bei allen Schnittstellendefinitionen zu finden.
Im folgenden werden die grundlegenden Mechanismen der verschiedenen allgemeinen Dienste beschrieben.

8.5. Aufbau der Systembibliotheken

Treiber sind Software-Bibliotheken zur Ansteuerung von Hardware-Schnittstellen. Die Treiber werden in der Regel in Form von dynamischen Bibliotheken (DLL) bereitgestellt.

Format der Funktionen

Der Funktionsprototyp aller M5dll-Funktionen ist ähnlich aufgebaut:
Funktionsname.
Der Name jeder Funktion - ausgenommen wenige allgemeine Funktionen, die in jedem Treiber vorhanden sind - beinhaltet die Thematik, der diese Funktion zuzuordnen ist.
Parameter und gelieferte Daten.
Von der Funktion zu liefernde Daten werden über Referenzparameter an das Anwenderprogramm übermittelt. Dadurch können Funktionen auch mehrere Informationen gleichzeitig bereitstellen.
Fehlercodes.
Der eigentliche Rückgabewert der Funktion stellt einen Fehlercode dar. Dadurch läßt sich der Erfolg des Funktionsaufrufs oder die genaue Ursache eines Fehlers feststellen. Es gilt, daß alle Werte ungleich 0 einen Fehlerzustand darstellen. Liefern Funktionen 0 zurück, so ist davon auszugehen, daß die Funktion erfolgreich ausgeführt wurde.

8.6. Programmieren mit der m5dll-Bibliothek

In diesem Abschnitt werden allgemein im Umgang mit der m5dll Bibliothek häufig benötigte Datentypen und Strukturen erläutert.

Allgemeines

Die m5dll-Bibliothek wird in der Regel in Form einer DLL ausgeliefert.
Folgende Dateien werden im Zusammenhang mit der Programmierung benötigt:
die Funktionsbibliothek (DLL) der m5dll,

C-*Header* -Dateien mit allgemeinen Definitionen, Konstanten und Funktionsprototypen.
Da meist C- oder C++-Programmierumgebungen benutzt werden, bezieht sich die Beschreibung der Datentypen und Strukturen auf die Programmiersprache C.

Dateien

Wie bereits oben genannt, werden neben der eigentlichen Bibliothek-DLL die C-Header-Dateien zur Programmentwicklung benötigt.
Diese Header-Dateien (*include* -Dateien) enthalten folgende Informationen:

- Definition der Standard-Datentypen
- Definition der verwendeten Strukturen
- Definition von Konstanten und Fehlercodes
- Liste der Funktionsprototypen

Die Header-Dateien beschreiben in der Regel keine Eigenschaften der konkreten Bibliothek, sondern die für mehrere Implementierungen gültige Schnittstelle. Daher werden für die konkrete Version der Bibliothek keine weiteren Informationen benötigt.

Konventionen

Zu den im folgenden genannten Konventionen zur Programmierung gehören die Datentypen und die Definition der Fehlercodes.

Datentypen

Die bei den diversen Funktionen verwendeten Standard-Datentypen sind:

Datentyp	Datentypbedeutung
Void	void
Char, Int8, Bool	8 Bit ganzzahlig mit Vorzeichen
UChar, Byte, UInt8	8 Bit ganzzahlig ohne Vorzeichen
Short, Int16	16 Bit ganzzahlig mit Vorzeichen
UShort, SWord, UInt16	16 Bit ganzzahlig ohne Vorzeichen
Int, Int32	32 Bit ganzzahlig mit Vorzeichen
UInt, Word, UInt32	32 Bit ganzzahlig ohne Vorzeichen
Float	32 Bit Gleitkomma
Double	64 Bit Gleitkomma
LongDouble	80 Bit Gleitkomma

Tabelle 49

Alle oben genannten Datentypen stellen lediglich speziell vereinbarte Namen für die in der Regel bereits in der Programmierumgebung vorhandenen Standarddatentypen dar. Diese Vereinbarungen befinden sich in der Datei TYPES.H. Darüberhinaus sind noch weitere Datentypen in dieser Datei vereinbart, die eine spezielle Verwendung bezeichnen.
Der von allen M5-Funktionen gelieferte Rückgabewert ist vom Typ:
int 32 Bit ganzzahlig mit Vorzeichen - wird verwendet, um den Erfolg eines Funktionsaufrufes anzuzeigen.

Rückgabewerte

Rückgabewert	Rückgabewert-Bedeutung
CLD_OK	Erfolg - kein Fehler
CLERR_INITIALIZATIONERROR	Fehler bei der Initialisierung
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_BADPARAM	falsche Parameter wurden an die Funktion übergeben
CLERR_NOINITSTRING	es wurde kein Initalisierungsstring angegeben

Tabelle 50

CLERR_NODEVICENAME	Schnittstelle wurde nicht angegeben
CLERR_BADDEVICEINITSTRING	Der angegebene Initalisierungsstring ist nicht gültig
CLERR_HARDWARENOTFOUND	Treiber findet keine Interfacekarte
CLERR_DEVICENOTOPEN	Schnittstelle konnte nicht geöffnet werden
CLDERR_THREADNOTCREATED	interner Thread konnte nicht erzeugt werden
CLDERR_WRONGHANDLE	Sie verwenden einen bereits gelöschten, oder noch nicht angelegten Zeiger auf ein Device
CLDERR_RECEIVEERROR	Angeforderter Parameter konnte nicht empfangen werden
CLDERR_DRIVE_NOMODULE-FOUND	Es wurden keine ansprechbaren Module gefunden
CLDERR_DRIVE_MODULETIME-OUT	Modul(e) antworten nicht

Tabelle 50

Die spezieller Thematik zugeordneten Fehlercodes sind im Referenzteil des vorliegenden Handbuches zu jeder Funktion erklärt.

Fehler

Falls eine PCube-Funktion einen Rückgabewert ungleich 0 liefert, wird damit ein Fehler signalisiert. Der Rückgabewert stellt eine 32-Bit-Konstante dar, die zur Fehleranalyse ausgewertet werden kann.

8.7. Grundlagen der m5dll-Schnittstelle

In diesem Abschnitt wird die Programmier-Schnittstelle *m5dll* zur Ansteuerung von intelligenten Antriebsmodulen beschrieben. Dazu zählen Funktionen zur Initialisierung der Schnittstelle zu den Antrieben, Funktionen zur Parametrierung der Antriebe und Bewegungsfunktionen. Die Funktionen der Programmierschnittstelle beziehen sich auf einen einzelnen Antrieb (*Drive*) der zu einer definierten Schnittstelle (Device) gehört.

Bewegungsart

Die modulspezifischen Funktionen zur Parametrierung und Bewegung erhalten in der Regel Werte im Gleitkommaformat (32 Bit), die die Längenangaben (bei Linearmodulen) oder Winkelangaben (bei Drehmodulen) in Einheiten des SI-Systems darstellen (m oder rad). Dadurch wird der Anwender von der sonst erforderlichen Umrechnung in sogenannte Inkremente (Geberimpulse der Antriebseinheiten) entlastet.
Der folgende Abschnitt beschreibt die erforderlichen Maßnahmen, um die Module zu initialisieren.

8.8. Initialisierung der verwendeten Schnittstelle (Device)

Für das Versenden und Empfangen von Nachrichten zwischen den intelligenten Antriebsmodulen und dem PC ist in der Regel eine Einsteckkarte verantwortlich. Sie muß zu Beginn der Arbeiten initialisiert und nach Abschluß entsprechend deinitialisiert werden. Dazu dienen die Funktionen PCube_openDevice und PCube_closeDevice. Mit dem zu übergebenden Initialisierungsstring erhält die m5dll die Information, auf welcher Schnittstelle die Kommunikation mit den Modulen stattfinden soll:
Zur Zeit werden folgende Schnittstellen unterstützt:

- VCI Interface der Firma IXXAT, CAN-Controller
- RS232

Der Initialisierungsstring besteht im groben aus zwei Teilen:

- der Schnittstellenkennung und den
- Parameterlisten.

Einige Parameter sind für alle Schnittstellen identisch, andere jedoch spezifisch für die

verwendete Schnittstelle.
Alle relevanten Informationen der Schnittstellenkarte müssen bekannt sein. Dies wären bei CAN-Karten z. B. die IO-Adresse , Interrupt, Kartentyp (PCMCIA, ISA, PCI) oder bei RS232 die Comport-Nummer und die Baudrate der Module (in der Regel 9600 Baud).

8.9. Der Initialisierungsstring

Der Initialisierungsstring bestimmt die verwendete Hardware, die vom Anwendungsprogramm angesteuert werden soll. Der Aufbau des Initialisierungsstrings ist aufgrund der Vielfalt der möglichen Hardwareeigenarten und variantenreichen Sonderfunktionen der m5dll recht komplex. Die Hardware unterstützt die Schnittstellen RS232, RS485, Profibus und CAN. Die Software unterstützt derzeit nur die Schnittstellen RS232 und CAN. Außerdem wird eine Softwareanbindung der Vorgänger-Baureihe (MORSE / M3DLL) ermöglicht.

Aufbau

Grundsätzlich ist die Form des Initialisierungsstring folgende:
SCHNITTSTELLENKENNUNG:par1,par2,par3,option1=on,option2=off"
Der Basis-Initialisierungsstring enthält die Schnittstellenkennung und bis zu drei Parameter. Weitere Optionen werden entsprechend angefügt.
Mögliche Schnittstellenkennungen und deren zugehörige Parameter sind:

Schnittstelle	Parameter			
VCI: oder M3VCI:	TYPE	Name	PORT bzw. PCMCIA Slot bzw. Board-Zähler, bzw. Basisadresse	Interrupt oder Baudrate (*)
	0	iPC-I 165	Basisadr.:z.B. d000	IRQ:1 -15
	1	iPC-I 320	Basisadr.:z.B. d000	IRQ:1 -15
	2	CANdy 320	LPT-Port z.B. 1	IRQ: 5 oder 7 bzw. der vom System für den LPT vergebene IRQ
	3	tinCAN V2	PCMCIA-Slot: z.B. 0	IRQ:1 -15
	4	keine Unterstützung		
	5	keine Unterstützung		
	6	iPC-I 165 PCI	Board-Nr.: z.B. 0	1
	7	iPC-I 320 PCI	Board-Nr.: z.B. 0	1
NET	HOST-IP-Adresse oder HOSTNAME : z.B. localhost oder 192.168.1.1			
RS232			COM Port: 1-4	BAUD Rate: z.B. 9600

Tabelle 51

(*) Bei Verwendung von Karten, die einen Interrupt (IRQ) unterstützen, sollte dieser in jedem Fall auch benutzt werden. Dies ist insbesondere bei der Verwendung von PowerConfig (Modulkonfiguration) zwingend notwendig!

Schnittstelle	Beispiel INIT-Strings

Tabelle 52

VCI: oder M3VCI:	TYPE	INIT-String
	0	VCI:0,D000,5 (Win 95/98 VCI:0,D000,5,nonet)
	1	VCI:1,D800,7 (Win 95/98 VCI:1,D800,7,nonet)
	2	VCI:2,1,7 (Win 95/98 VCI:2,1,7,nonet)
	3	VCI:3,0,5 (Win 95/98 VCI:3,0,5,nonet)
	4	keine Unterstützung
	5	keine Unterstützung
	6	VCI:6,0,1 (Win 95/98 VCI:6,0,1,nonet)
	7	VCI:7,0,1 (Win 95/98 VCI:7,0,1,nonet)
NET	NET: LOCALHOST NET: 192.168.1.1	
RS232	RS232:1,9600 (Win 95/98 RS232:1,9600,nonet)	
Profibus IM180	IM180:0,0 (Win 95/98 IM180:0,0,nonet)	

Tabelle 52

```
char* pInitString = "VCI:1,d000,11";
// Startadresse d000H
// Interrupt 11
// VCI-CAN Interface
// Type iPC320
int xtrnlID;
int retVal = Pcube_openDevice( &xtrnlID, pInitString );
if( retVal != CLD_OK ) // Initialisierung fehlerhaft?
... // ja - Fehlerauswertung
else
{ ... // weitere Funktionen
}
Pcube_closeDevice( xtrnlID ); // letzter Funktionsaufruf...
```

Optionen

Die im folgenden beschriebenen Optionen werden durch Komma getrennt nach dem Basisinitialisierungsstring angefügt. Folgende Optionen bestehen, um die m5dll zusätzlich zu parametrieren:

Debugmodus

Durch Anfügen der Option D direkt nach dem Basis-Initialisierungsstring wird die Kommunikation der m5dll mit den PowerCube™-Modulen in eine Datei debug.txt protokolliert. Diese Dateil liegt im Verzeichnis der Anwendung, die die m5dll verwendet.

Netzwerkoptionen

Die Optionen der Schnittstellenkennung NET: lauten:
port=Portadresse Falls ein Server gestartet wurde, der eine Netzwerkfunktionalität besitzt, so wird mittels dieser Option dessen Portaddresse gesetzt. Diese muß der des Servers entsprechen.
passwd=PASSWORT Falls ein Server gestartet wurde der eine Netzwerkfunktionalität besitzt, so wird mittels dieser Option ein Authentifizierungspasswort gesetzt. Dieses muß dem des Servers entsprechen.

CAN-Optionen

Die Optionen der Schnittstellenkennung VCI: und M3VCI: lauten:
nonet Es soll kein Netzwerkserver gestartet werden.
port=Portadresse Falls ein Netzwerkserver gestartet werden soll wird mittels dieser Option dessen Portaddresse gesetzt
passwd=PASSWORT Falls ein Server gestartet werden soll, so wird mittels dieser Option ein Authentifizierungspasswort gesetzt.

RS232-Optionen

nonet Es soll kein Netzwerkserver gestartet werden.
port=Portadresse Falls ein Netzwerkserver gestartet werden soll wird mittels dieser Option dessen Portaddresse gesetzt
passwd=PASSWORT Falls ein Server gestartet werden soll, so wird mittels dieser

Option ein Authentifizierungspasswort gesetzt.

8.10. Allgemeine Sonderfunktionen

Anzahl der Module

Mittels der Funktion PCube_getModuleCount kann ermittelt werden, wie viele Module an der spezifizierten Schnittstelle angeschlossen sind.
Mit Hilfe der Funktion PCube_getModulesIdMap kann eine Liste der gefundenen Module mit den physikalischen ID's erzeugt werden. Diese Funktion liefert darüber hinaus noch die Anzahl der gefundenen Module zurück.

```
int idMap[30];
numDrives=Pcube_getModulesIdMap(xtrnlId,idMap);
```

Modultyp

Zur Feststellung der Bewegungsform des jeweiligen Antriebsmoduls und anderer wesentlicher Eigenschaften dient die Funktion PCube_getModuleType. Die Funktion liefert den Typ des Moduls zurück und entscheidet über die Interpretation der Funktionsparameter. Folgende Modultypen sind definiert:
ROTARY_DRIVE Antriebsmodul ist ein Drehmodul, die Parameter werden in der Einheit rad (Radiant) angegeben.
LINEAR_DRIVE Antriebsmodul ist ein Linearmodul, die Parameter werden in der Einheit m (Meter) angegeben.

```
UInt8 type;
CLDRet retVal = Pcube_getModuleType( xtrnlId, &type );
if(type == LINEAR_DRIVE )
... // Angaben in Meter
else if( type == ROTARY_DRIVE )
... // Angaben in Radiant
... // weitere Funktionen
```

8.11. Programmierung von Bewegungen

Im Folgenden werden die Einheiten der Bewegungsparameter sowie die Bedeutung einzelner Funktionskategorien erläutert. Zur Verwendung der Funktionen siehe auch Abschnitt 15.8, '*Referenz* '.

Einheiten

Bei der Programmierung von Bewegungen und der Einstellung von Bewegungsparametern haben alle Angaben in SI-Einheiten zu erfolgen, um weiterführender Berechnungen zu vereinfachen.
Die verwendeten Einheiten sind:

Einheiten	Linearantrieb	Drehantrieb
Position	m	rad
Geschwindigkeit	m/s	rad/s
Beschleunigung	m/s^2	rad/s^2
Motorstrom	A	A
Temperatur	K	K

Tabelle 53

Verfahrbereich

Das Betriebssystem des Antriebsmoduls schränkt normalerweise den möglichen Verfahrbereich ein. Dieser *Verfahrbereich* bezeichnet den Abstand zwischen den im

Modul eingestellten Softwareendlagen.

Dynamik

Bei allen Bewegungen des Antriebs werden voreingestellte oder von der übergeordneten Steuerung vorgegebene Werte für Fahrgeschwindigkeit und -beschleunigung verwendet. Dabei werden zwei auch als *Dynamik* bezeichnete Wertepaare benutzt. Die *aktuelle* Dynamik wird bei allen von der übergeordneten Steuerung (dem PC-Anwenderprogramm) ausgelösten Bewegungen verwendet. Die *maximale* Dynamik dient dem Modul-Betriebssystem zur Begrenzung aller Bewegungen. Diese Grenzwerte sind standardmäßig im Modul voreingestellt. Von der übergeordneten Steuerung können sie aber weiter eingeschränkt werden. Diese Parametereinstellungen sind online möglich. Nach Abschalten sind die Werte jedoch verloren.

Status

Bit im Statuswort	Bedeutung
STATE_HOME_OK	Die Homing-Fahrt des Moduls wurde bereits erfolgreich durchgeführt.
STATE_HALTED	Der HALT-Zustand wurde aufgrund eines weiteren Fehlers oder durch das Kommando Pcube_haltModule() hervorgerufen.
STATE_SWR	Das Modul befindet sich im Bereich des Referenzschalters.
STATE_SW1	Das Modul befindet sich im Bereich des Endlagenschalters 1.
STATE_SW2	Das Modul befindet sich im Bereich des Endlagenschalters 2
STATE_MOTION	Das Modul befindet sich in einer Bewegung
STATE_RAMP_ACC	Das Modul beschleunigt gerade.
STATE_RAMP_STEADY	Das Modul befindet sich in einer Rampenfahrt im Bereich mit konstanter Geschwindigkeit.
STATE_RAMP_DEC	Das Modul bremst ab (absteigende Rampenfahrt).
STATE_RAMP_END	Die Rampenfahrt wurde beendet.
STATE_INPROGRESS	Ein Step-Kommando ist in Ausführung. Es kann noch eine weitere Position mittels Pcube_moveStep() übergeben werden (nicht bei M3-Kompatibilität).
STATE_FULLBUFFER	Das zuletzt empfangene STEP-Kommando wird in den Puffer übernommen und nach Ausführung des vorhergehenden STEP-Kommandos automatisch "angehängt" (nicht bei M3-Kompatibilität).
STATE_BRAKEACTIVE	Die Bremse ist aktiviert.
STATE_CURLIMIT	Der Antrieb befindet sich mit den aktuellen Einstellungen im Grenzlastbereich
STATE_ERROR	Kennzeichnet, daß ein Fehler vorliegt. Die Fehlerursache kann durch Auswertung der restlichen Flags bestimmt werden.
STATE_POWERFAULT	Ein Fehler in der Motorendstufe des Moduls wurde entdeckt.
STATE_TOW_ERROR	Die maximale Abweichung von der Sollposition wurde überschritten.
STATE_COM_ERROR	Ein Fehler in der Kommunikation wurde bemerkt.
STATE_POW_VOLT_ERR	Die Zwischenkreisspannung des Motors ist unter den zulässigen Minimalwert gesunken. Die Endstufe wird abgeschaltet.
STATE_POW_FET_TEMP	Die Temperatur der Endstufentransistoren hat das zulässige Maximum überschritten. Die Endstufe wird abgeschaltet.
STATE_POW_WDG_TEMP	Die Temperatur der Motorwicklungen hat das zulässige Maximum überschritten. Die Endstufe wird abgeschaltet.
STATE_POW_SHORTCUR	Es ist ein Kurzschluß in der Endstufe aufgetreten. Die Endstufe wird abgeschaltet
STATE_POW_HALLERR	Es ist ein Fehler bei der Erfassung der Hallimpulse aufgetreten. Die Endstufe wird abgeschaltet.

Tabelle 54

STATE_POW_INTEGRALERR	Die Dauerbelastung des Motors hat die Grenze der Zulässigkeit überschritten. Die Endstufe wird abgeschaltet.
STATE_CPU_OVERLOAD	Die CPU ist überlastet Die Endstufe wird abgeschaltet.

Tabelle 54

Für die genaue Beschreibung der Statusbits sowie der notwendigen Reaktionen darauf ist unbedingt das Kapitel "Das PowerCube™ -Betriebssystem", Abschnitt Der Modulstatus: Das Statuswort CubeState" zu lesen.

8.12. Funktionsübersicht

Zur Vereinfachung der Programmierung genügt es, wenn die Datei m5.H mittels der Anweisung `#include "m5.h"` eingefügt wird. Die Include-Datei muß sich in einem dem Projekt zugänglichen Verzeichnis befinden.
Im folgenden Referenzteil werden alle Funktionen erklärt. Ihre Parameter und Rückgabewerte sowie die Bedeutung des Funktionsaufrufes werden beschrieben.

Verwaltungsfunktionen

PCube_openDevice	Initialisierung der Treiber-DLL
PCube_closeDevice	Beenden der Treiber-DLL
PCube_getModulesCount	Abfrage der Anzahl der Drives
PCube_getModulesIdMap	Abfrage der ID-Zuordnung
PCube_getModuleType	Abfrage des Modultyps

Tabelle 55

Kommandofunktionen

PCube_syncModule	Referenzpunkt anfahren
PCube_haltModule	Antriebsmodul anhalten
PCube_resetModule	Rücksetzen des Modulstatus

Tabelle 56

Bewegung

PCube_moveRamp	Bewegungskommando im Rampenmodus
PCube_moveRampInc	Bewegungskommando im Rampenmodus (Parameter in Inkrementen)
PCube_moveRampExtended	Bewegungskommando im Rampenmodus mit Statusrückgabe
PCube_moveStep	Bewegungskommando im Schrittmodus
PCube_moveStepInc	Bewegungskommando im Schrittmodus (Parameter in Inkrementen)
PCube_moveStepExtended	Bewegungskommando im Schrittmodus mit Statusrückgabe
PCube_moveVel	Bewegungskommando im Geschwindigkeitsmodus
PCube_moveVelInc	Bewegungskommando im Geschwindigkeitsmodus (Parameter in Inkrementen)
PCube_moveCurrent	Bewegungskommando im Strommodus

Tabelle 57

PCube_moveCurrentInc	Bewegungskommando im Strommodus (Parameter in Inkrementen)
PCube_moveCurrentExtended	Bewegungskommando im Strommodus mit Statusrückgabe

Tabelle 57

Zustandsdaten Abfragen des aktuellen Modulzustands

PCube_getCubeState	allgemeine Funktion zur Statusabfrage
PCube_getActPos	aktuelle Position abfragen
PCube_getActVel	aktuelle Geschwindigkeit abfragen
PCube_getDeltaPos	aktuellen Schleppfehler der Position abfragen
PCube_getCur	aktuelle Stromaufnahme abfragen

Tabelle 58

Endlagen Abfragen und Setzen von Bewegungsgrenzwerten

PCube_getMaxPos	maximale Position abfragen
PCube_setMaxPos	maximale Position setzen
PCube_getMinPos	minimale Position abfragen
PCube_setMinPos	minimale Position setzen
PCube_getHomeOffset	benutzerdefinierte Nullposition abfragen
PCube_setHomeOffset	benutzerdefinierte Nullposition setzen
PCube_getMaxVel	maximale Geschwindigkeit abfragen
PCube_setMaxVel	maximale Geschwindigkeit setzen
PCube_getMinVel	minimale Geschwindigkeit abfragen
PCube_setMinVel	minimale Geschwindigkeit setzen
PCube_getMaxAcc	maximale Beschleunigung abfragen
PCube_setMaxAcc	maximale Beschleunigung setzen
PCube_getMinAcc	minimale Beschleunigung abfragen
PCube_setMinAcc	minimale Beschleunigung setzen
PCube_getMaxCur	maximalen Strom abfragen
PCube_setMaxCur	maximalen Strom setzen
PCube_getMinCur	minimalen Strom abfragen
PCube_setMinCur	minimalen Strom setzen
PCube_getMaxDeltaPos	maximale Positionsabweichung abfragen
PCube_setMaxDeltaPos	maximale Positionsabweichung setzen

Tabelle 59

Reglerkoeffizienten Abfragen und Setzen der Koeffizienten des Lageregelkreises
Diese Funktionen stehen erst ab der Version 2.5.1 zur Verfügung

PCube_getC0	Koeffizient C0 abfragen

Tabelle 60

PCube_getDamp	Dämpfungswert abfragen
PCube_getA0	Koeffizient A0 abfragen
PCube_setC0	Koeffizient C0 setzen
PCube_setDamp	Dämpfungswert setzen
PCube_setA0	Koeffizient A0 setzen
PCube_recalcPIDParams	Neuberechnen der Regelschleife

Tabelle 60

Digital I/O Abfragen und Setzen der internen Digital-Ein- und Ausgänge

PCube_getDioData	Bits der internen DIOs abfragen
PCube_setDioData	Bits der internen DIOs setzen

Tabelle 61

Defaultwerte Abfrage von Vorgabewerten

PCube_getDefMaxPos	maximale Position abfragen
PCube_getDefMinPos	minimale Position abfragen
PCube_getDefMaxVel	maximale Geschwindigkeit abfragen
PCube_getDefMinVel	minimale Geschwindigkeit abfragen
PCube_getDefMaxAcc	maximale Beschleunigung abfragen
PCube_getDefMinAcc	minimale Beschleunigung abfragen
PCube_getDefMaxCur	maximalen Strom abfragen
PCube_getDefMinCur	minimalen Strom abfragen
PCube_getDefDeltaPos	maximale Positionsabweichung abfragen
PCube_getDefC0	Koeffizient C0 abfragen
PCube_getDefDamp	Dämpfung abfragen
PCube_getDefA0	Koeffizient A0 abfragen

Tabelle 62

8.13. Initialisierungs- und Verwaltungsfunktionen

Die im Folgenden beschriebenen Funktionen dienen der Initialisierung der Schnittstelle und des angeschlossenen Manipulators mit seinen Antriebsmodulen.

8.14. PCube_openDevice

Aufgabe Initialisierung der Schnittstelle zum PowerCube Drive

Prototyp

```
int PCube_openDevice( int* pDev, const char* pInitString );
```

Beschreibung Diese Funktion initialisiert die Schnittstelle die anhand der in der Zeichenkette *pInitString* übergebenen Parameter. Der Aufbau des Initialisierungsstrings hängt von der Installation Schnittstelle ab. Der Parameter pDev ist bei allen weiteren Pcube Funktionen anzugeben. Es ist möglich, mehrere verschiedene Schnittstellen innerhalb einer Anwendung zu öffnen. Die Identifikation, welche Schnittstelle verwendet wird, ist

durch das zurückgelieferte Handle pDev bekanntgegeben.

Rückgabewert

CLD_OK	Schnittstelle konnte erfolgreich initialisiert werden.
CLDERR_INITILIZATIONERROR	Fehler bei der Initialisierung
CLDERR_BADPARAM	Falsche Parameter im Initialisierungsstring oder falsch formatiert (siehe entsprechenden Abschnitt im Anhang).
CLDERR_NOINITSTRING	es wurde kein Initalisierungsstring angegeben
CLDERR_NODEVICENAME	Schnittstelle wurde nicht angegeben
CLDERR_BADDEVICEINITSTRING	Der angegebene Initalisierungsstring ist nicht gültig
CLDERR_HARDWARENOTFOUND	VCI-Treiber findet keine Interfacekarte
CLDERR_DEVICENOTOPEN	Schnittstelle konnte nicht geöffnet werden
CLDERR_THREADNOTCREATED	interner Thread konnte nicht erzeugt werden
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 63

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.15. PCube_closeDevice

Aufgabe Initialisierung der Schnittstelle zum PowerCube Drive

Prototyp

```
int PCube_closeDevice( int dev);
```

Beschreibung Diese Funktion gibt die von der geöffneten Schnittstelle dev belegten Ressourcen frei. Des weiteren wird ein HALT-Kommando an alle an der definierten Schnittstelle vorhandenen Module geschickt.

Rückgabewert

CLD_OK	Device wurde erfolgreich freigegeben
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht

Tabelle 64

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.16. PCube_getModulesCount

Aufgabe Abfrage der Anzahl Module

Prototyp

```
int PCube_getModulesCount(int dev, int* count);
```

Beschreibung Über den Zeiger *count* wird die Zahl der erkannten Antriebsmodule an das Anwenderprogramm zurückgeliefert.

Rückgabewert

CLD_OK	Es wurde erfolgreich eine Kommunikation durchgeführt...
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 65

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.17. PCube_getModulesIdMap

Aufgabe Abfrage der Drive ID's eines Devices

Prototyp

```
int PCube_getModulesIdMap(int dev, int *idMap);
```

Beschreibung Diese Funktion füllt eine Liste mit den gefundenen ID's der angeschlossenen Module. Somit ist ein Ansprechen der Module mittels logischer Nummern möglich, ohne die ID's der einzelnen Drives explizit zu kennen.

Rückgabewert

retVal	Anzahl der gefundenen Module
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 66

8.18. PCube_getModuleType

Aufgabe Abfrage der Drive ID's eines Devices

Prototyp

```
int PCube_getModuleType(int dev, int cube, UInt8 *type);
```

Beschreibung Die Funktion fordert den Modultyp des Antriebsmoduls *cube* ab. Dies ist ein 8-Bit-Wort, das in der Variablen abgelegt wird, auf die der Zeiger *type* zeigt.
Es sind zwei Grundtypen definiert:

ROTARY_DRIVE 0x0f	Angaben in Radiant
LINEAR_DRIVE 0xf0	Angaben in m

Rückgabewert

CLD_OK	Es wurde erfolgreich eine Kommunikation durchgeführt...
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 67

8.19. Kommandofunktionen

Die folgenden Funktionen stellen komplexe Kommandos dar, die das Antriebsmodul selbständig ausführt.

8.20. PCube_syncModule

Aufgabe

Referenzpunkt anfahren

Prototyp

```
int PCube_syncModule( int dev,int cube);
```

Beschreibung

Mit Aufruf dieser Funktion fährt das Antriebsmodul *cube* automatisch seinen Referenzpunkt an.
Vor dem ersten Fahrkommando muß das Antriebsmodul synchronisiert werden. Das heißt, es muß seinen Referenzpunkt anfahren, um einen absoluten Bezug zum umgebenden Koordinatensystem zu erhalten.
Die Funktion PCube_syncModule startet den Vorgang des selbständigen Anfahrens des Referenzpunktes. Mit Hilfe der Funktion PCube_querySyncEnd kann vom Anwenderprogramm aus der Abschluß des Vorgangs festgestellt werden.
Nach dem erfolgreichen Anfahren des Referenzpunktes werden Bewegungskommandos erst wirksam, wenn der Sicherheitszustand im Modul aufgehoben wurde (siehe PCube_resetModule).

Rückgabewert

CLD_OK	Anfahren des Referenzpunktes konnte erfolgreich begonnen werden.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 68

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

Beispiel

Das folgende vereinfachte Beispiel demonstriert das Anfahren des Referenzpunktes unter Nutzung einer Programmschleife
Das Anfahren des Referenzpunktes kann mehrere Sekunden dauern. Es sollte in solch einem Anwendungsprogramm ein zusätzlicher Thread erzeugt werden, der das korrekte Referenzpunktfahren übernimmt und die Anwendung nicht behindert.

```
UInt32 state;
CLDRet retVal = PCube_syncModule( dev, mod );
if( retVal == CLD_OK )
{ Bool sync = FALSE;
  ...
}
while( !sync )
{ Sleep(100);
  retVal = PCube_getCubeState( dev, mod, &state );
  if( (state & STATE_HOME_OK) && (state & STATE_RAMP_END) )
    sync = TRUE;
}
```

Die Endlosschleife wird abgebrochen, wenn ein Fehler auftritt oder der Referenzpunkt erreicht wurde.

Beispiel

Folgendes Beispiel zeigt das Anfahren des Referenzpunktes mit Hilfe von WM_TIMER-Nachrichten unter Windows.

```
CLDRet retVal;
Bool sync;
...
case WM_COMMAND:
  ... // Referenzpunkt anfahren
  retVal = PCube_syncModule( dev, i );
  ...
```

```
  break;
case WM_TIMER:
  ... // Referenzpunkt erreicht?
  sync = FALSE;
  retVal = PCube_querySyncEnd( dev, i, &sync );
  if( retVal != CLD_OK )
    ... // FEHLER!
  else if( sync )
    ... // OK, Modul synchronisiert.
  else
    ... // Vorgang dauert noch an ...
  break;
```

Die Nachricht WM_TIMER sollte solange den Vorgang überwachen - beispielsweise im zeitlichen Abstand von 200 Millisekunden - bis ein Fehler festgestellt oder der Referenzpunkt erreicht wurde. Durch diese Trennung des Vorgangs in Starten und zyklisches Überprüfen wird das System nicht unnötig lange blockiert.
Siehe auch PCube_querySyncEnd, PCube_resetModule

8.21. PCube_haltModule

Aufgabe Antriebsmodul anhalten

Prototyp

```
int PCube_haltModule( int dev,int cube);
```

Beschreibung Diese Funktion bewirkt das automatische Anhalten des Antriebsmoduls *cube* .
Nach der Ausführung dieser Funktion geht das Antriebsmodul automatisch in einen Sicherheitszustand über. In diesem Zustand werden vom Modul keine Bewegungskommandos angenommen. Zur Aufhebung dieses Zustands muß das Kommando PCube_resetModule gesendet werden.
Die Funktion PCube_haltModule startet den Vorgang des selbständigen Anhaltens des Antriebsmoduls. Mit Hilfe der Funktion PCube_queryEndPos kann vom Anwenderprogramm aus der Stillstand des Moduls festgestellt werden.

Rückgabewert

CLD_OK	Anhalten des Moduls konnte erfolgreich begonnen werden.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 69

Bei anderen Rückgabewerten siehe ‘Vorgehensweise bei Fehlern ’.
Siehe auch PCube_queryEndPos, PCube_resetModule

8.22. PCube_resetModule

Aufgabe Rücksetzen des Modulstatus

Prototyp

```
int PCube_resetModule( int dev,int cube);
```

Beschreibung Diese Funktion bewirkt die Aufhebung des Sicherheitszustandes im Antriebsmodul *cube* und erlaubt damit Bewegungskommandos.
Folgende Ereignisse führen zur Einnahme des Sicherheitszustands im Antriebsmodul:

- Feststellung eines Fehlers,
- Auslösung eines HALT-Kommandos durch das Anwenderprogramm

(durch Aufruf der Funktion PCube_haltModule)
Erst nach Ausführung der Funktion PCube_resetModule werden vom Antriebsmodul Bewegungskommandos angenommen. Die Funktion PCube_resetModule scheitert, wenn noch die Bedingungen vorliegen, die zu einem Fehler führten. Mit Hilfe der Funktion PCube_getCubeState kann vom Anwenderprogramm aus der Status des Moduls festgestellt werden.
Mögliche Fehler sind Im Kapitel "Beschreibung des PowerCube‘ - Kommunikationsprotokolls", Abschnitt "Der Modulstatus: Das Statuswort CubeState" aufgeführt.

Rückgabewert

CLD_OK	RESET-Kommando wurde erfolgreich an das Modul gesendet
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 70

Bei anderen Rückgabewerten siehe ‘*Vorgehensweise bei Fehlern*’.
Siehe auch PCube_syncModule, PCube_haltModule, PCube_getCubeState

8.23. Statusfunktionen

Die im Folgenden genannten Funktionen beschreiben den Status des Antriebsmoduls.

8.24. PCube_getCubeState

Aufgabe Allgemeine Funktion zur Statusabfrage

Prototyp
```
int PCube_getCubeState( int dev, int cube, UInt32* pValue );
```

Beschreibung Die Funktion fordert das Statuswort des Antriebsmoduls *cube* ab. Dies ist ein 32-Bit-Wort, das in der Variablen, auf die der Zeiger *pValue* zeigt, abgelegt wird. Jedes Bit in diesem Statuswort hat eine bestimmte Bedeutung, die weiter unten erklärt wird.
Hinweis: Das Statuswort ist im Abschnitt Status des Antriebsmoduls abfragen beschrieben

Rückgabewert Bei anderen Rückgabewerten siehe ‘*Vorgehensweise bei Fehlern*’.

CLD_OK	Status des Moduls wurde abgefordert.
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 71

Siehe auch PCube_querySyncEnd, PCube_queryEndPos

8.25. PCube_querySyncEnd

Aufgabe Prüfung, ob Referenzpunkt erreicht

Prototyp
```
int PCube_querySyncEnd( int dev, int driveId, Bool* pBool );
```

Beschreibung

Die Funktion vereinfacht die Prüfung auf Erreichen des Referenzpunktes. Sie ist nur sinnvoll nach Ausführung der Funktion PCube_syncModule. Das Ergebnis der Prüfung (TRUE = = Referenzpunkt er-reicht) wird über pBool zurückgeliefert.

Rückgabewert

CLD_OK	Prüfung ist erfolgt - in der Variablen, auf die der Zeiger *pBool* zeigt, befindet sich das Ergebnis der Prüfung.
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 72

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_syncModule

8.26. PCube_queryEndPos

Aufgabe

Prüfung, ob Endpunkt der Bewegung erreicht

Prototyp

```
int PCube_queryEndPos( int dev, int driveId, Bool* pBool );
```

Beschreibung

Die Funktion vereinfacht die Prüfung auf Erreichen des Endpunktes einer Rampenbewegung. Sie ist nur sinnvoll nach Ausführung der Funktion PCube_moveRamp. Darüberhinaus wird nach Auslösung des HALT-Kommandos oder nach Auftreten eines Fehlers im Moduls angezeigt, ob das Modul seinen Stillstand erreicht hat. Das Ergebnis der Prüfung (TRUE = = Endpunkt erreicht) wird über *pBool* zurückgeliefert.

Rückgabewert

CLD_OK	Prüfung ist erfolgt - in der Variablen, auf die der Zeiger *pBool* zeigt, befindet sich das Ergebnis der Prüfung.
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 73

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Beispiel Ein ausführliches Beispiel ist zur Funktion PCube_moveRamp zu finden.
Siehe auch PCube_moveRamp

8.27. Bewegungsfunktionen

Bei Verwendung von Greifer-Modulen (PG70, PG90) ist auf folgende Besonderheit zu achten. Wird dieses Modul nicht im Current-Modus bewegt (PCube_moveCurrent), ist unbedingt darauf achten, daß die eingegebene Endposition mit der Außenposition des zu greifenden Gegenstandes übereinstimmt. Ansonsten erfolgen Schleppfehlermeldungen.

8.28. PCube_moveRamp

Aufgabe

Bewegungskommando im Rampenmodus

Prototyp

```
int PCube_moveRamp( int dev, int cube, float pos, float vel, float acc);
```

Beschreibung

Die Funktion löst eine Bewegung des Antriebsmoduls *cube* im Rampenmodus aus. Dazu wird die Zielposition *pos* mit der maximalen Geschwindigkeit *vel* selbständig angefahren. Die Beschleunigung erfolgt mit dem in *acc* angegebenen Wert. Falls sich das Modul zum Zeitpunkt des Funktionsaufrufes bereits in einer Bewegung befand, wird mit der angegebenen Dynamik unmittelbar die neue Zielposition angefahren.
Die Zielposition *pos* bezeichnet den Abstand zum Referenzpunkt. Die Werte *pos* , *vel* und *acc* werden in Einheiten des SI-Systems angegeben (siehe 'Einheiten'). Werden höhere Maximalwerte, als die aktuell zulässigen angegeben, erfolgt eine automatische Begrenzung auf die Maximalwerte. Ein Überfahren der Endlagen ist nicht möglich.
Die Funktion schlägt fehl, wenn
das Modul noch nicht seinen Referenzpunkt angefahren hatte und zurückgesetzt wurde (siehe PCube_syncModule und PCube_resetModule),
sich das Modul nach einem Fehler im Sicherheitszustand befindet (siehe PCube_getCubeState)
Das bedeutet, die Funktion wurde scheinbar korrekt ausgeführt, das Antriebsmodul bewegt sich jedoch nicht bis zur Zielposition. Aus diesem Grund ist es ratsam, vor Ausführung der Funktion den Modulstatus zu überprüfen (Funktion PCube_getCubeState). Bei folgendem Bitmuster ist die Funktion korrekt ausführbar, das heißt die Bewegung wird (zumindest bis zur Endlagenposition) stattfinden:

Bit im Statuswort	erforderlicher Zustand
STATE_HALTED	darf nicht gesetzt sein
STATE_HOME_OK	muß gesetzt sein
STATE_ERROR	darf nicht gesetzt sein

Befindet sich die geforderte Zielposition außerhalb der Softwareendlagen, wird das Modul entsprechend der Funktionsparameter soweit es die Softwareendlagen zulassen bewegt. Unmittelbar vor Erreichen der Softwareendlagen wird das Modul mit der programmierten Beschleunigung abgebremst. Die vorgegebene Endlage wird somit nicht überschritten.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 74

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern* '.

Beispiel

```
Float pos = 0.120, vel = 0.045, acc = 0.100;
Bool ready;
...
case WM_COMMAND:
... // Zielposition anfahren
retVal = PCube_moveRamp( dev, driveId, pos, vel, acc );
...
break;
case WM_TIMER:
... // Endposition erreicht?
ready = FALSE;
```

```
retVal = PCube_queryEndPos( dev, driveId, &ready );
if( retVal != CLD_OK )
... // FEHLER!
else if( ready )
... // OK, Zielposition erreicht.
else
... // Vorgang dauert noch an oder
// prüfen, ob Fehler oder
// Softwareendlage ...
```

Siehe auch PCube_queryEndPos, PCube_move...

8.29. PCube_moveRampInc

Aufgabe Bewegungskommando im Rampenmodus

Prototyp

```
int PCube_moveRampInc( int dev, int cube, Int32 pos, Int32 vel,
Int32 acc);
```

Beschreibung Die Funktion wird wie PCube_moveRamp eingesetzt. Der Unterschied besteht darin, daß hier Integer-Werte statt Float-Parameter übergeben werden können. Zur Dimensionierung der Parameter siehe unter Maßeinheiten der PowerCube™-Module.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 75

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_moveStepInc, PCube_moveVelInc, PCube_moveCurrentInc

8.30. PCube_moveRampExtended

Aufgabe Bewegungskommando im Rampenmodus mit erweiterter Statusrückgabe. Verfügbar ab PowerCube™-Betriebssystemversion 2.5.16 bzw. 3.5.13

Prototyp

```
int PCube_moveRampExtended( int dev, int cube, float pos, float
vel, float acc, unsigned long *stateRet, unsigned char *dioRet,
float *posRet);
```

Beschreibung Die Funktion wird wie PCube_moveRamp mit dem Unterschied eingesetzt, daß über die Zeiger stateRet, dioRet und posRet ein Kurzstatus, der Zustand der internen digitalen IOs und die aktuelle Position des Moduls zurückgeliefert werden und für eine sofortige Auswertung zur Verfügung stehen (siehe auch Kapitel 7.2.8, Befehlsbeispiele)

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 76

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_moveStepExtended, PCube_moveVelExtended, PCube_moveCurrentExtended

8.31. PCube_moveStep

Aufgabe Bewegungskommando im Schrittmodus.

Prototyp

```
int PCube_moveStep( int dev, int cube, float pos, UInt16 step);
```

Beschreibung Die Funktion bewegt das Antriebsmodul *cube* im Schrittmodus. Dabei wird versucht, die angegebene Zielposition *pos* innerhalb von *step* Millisekunden anzufahren. Die berechnete Geschwindigkeit ist: v = (*pos* - <aktuelle Position>) / *step*.
Die im Antriebsmodul integrierte Bewegungssteuerung nimmt selbständig eine eventuell erforderliche Beschleunigung auf die errechnete Geschwindigkeit vor. Hierfür werden die vorgegebenen maximalen Dynamik-Werte verwendet. Sofern die maximalen Dynamikwerte nicht überschritten werden, sorgt die Bewegungssteuerung für das Erreichen der Zielposition zum vorgegebenen Zeitpunkt. Selbst dann, wenn der Wert der resultierenden Geschwindigkeit kurzzeitig höher als der der Berechneten ist.
Falls spätestens bis Ablauf von *step* Millisekunden kein neues Schrittkommando oder ein Kommando einer anderen Betriebsart gesendet wurde, hält das Modul selbständig mit der aktuellen (Brems-) Beschleunigung an.
Die unter PCube_moveRamp vorgestellten Informationen über das Fehlschlagen der Funktion, über den erforderlichen Modulstatus und zum Verhalten vor Softwareendlagen gelten auch hier.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 77

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_move...

8.32. PCube_moveStepInc

Aufgabe Bewegungskommando im Schrittmodus

Prototyp

```
int PCube_moveStepInc( int dev, int cube, Int32 pos, UInt16 step);
```

Beschreibung Die Funktion wird wie PCube_moveStep mit dem Unterschied eingesetzt, daß hier Integer-Werte statt Float-Parameter übergeben werden können. Zur Dimensionierung der Parameter siehe unter Maßeinheiten der PowerCube™-Module.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 78

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_moveRampInc, PCube_moveVelInc, PCube_moveCurrentInc

8.33. PCube_moveStepExtended

Aufgabe

Bewegungskommando im Schrittmodus mit erweiterter Statusrückgabe. Verfügbar ab PowerCube™-Betriebssystemversion 2.5.16 bzw. 3.5.13

Prototyp

```
int PCube_moveStepExtended( int dev, int cube, float pos, unsigned short itime, unsigned long *stateRet, unsigned char *dioRet, float *posRet);
```

Beschreibung

Die Funktion wird wie PCube_moveStep mit dem Unterscheid eingesetzt, daß über die Zeiger stateRet, dioRet und posRet ein Kurzstatus, der Zustand der internen digitalen IOs und die aktuelle Position des Moduls zurückgeliefert werden und für eine sofortige Auswertung zur Verfügung stehen (siehe auch Kapitel 7.2.8, Befehlsbeispiele)

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 79

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_moveRampExtended, PCube_moveCurrentExtended, PCube_moveVelExtended

8.34. PCube_moveVel

Aufgabe

Bewegungskommando im Geschwindigkeitsmodus

Prototyp

```
int PCube_moveVel( int dev, int cube, float vel);
```

Beschreibung

Das Antriebsmodul *cube* wird in den Geschwindigkeitsmodus überführt. Das Modul beschleunigt mit der maximal zulässigen Beschleunigung selbständig auf die neue Geschwindigkeit *vel*. Die angegebene Geschwindigkeit *vel* darf nicht größer als die programmierte maximale Geschwindigkeit sein, ansonsten wird *vel* begrenzt.
Die Geschwindigkeit *vel* wird je nach Modultyp in m/s oder rad/s angegeben.
Das Modul bewegt sich kontinuierlich mit der angegebenen Geschwindigkeit, bis ein anderes Bewegungskommando oder ein HALT-Kommando gesendet wird oder bis die Softwareendlagen erreicht werden.
Die unter PCube_moveRamp vorgestellten Informationen über das Fehlschlagen der Funktion, über den erforderlichen Modulstatus und zum Verhalten vor Softwareendlagen gelten auch hier.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 80

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_move...

8.35. PCube_moveVelInc

Aufgabe Bewegungskommando im Geschwindigkeitsmodus

Prototyp
```
int PCube_moveVelInc( int dev, int cube, Int32 vel);
```

Beschreibung Die Funktion wird wie PCube_moveVel mit dem Unterschied eingesetzt , daß hier Integer-Werte statt Float-Parameter übergeben werden können. Zur Dimensionierung der Parameter siehe unter Maßeinheiten der PowerCube™-Module.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 81

Bei anderen Rückgabewerten siehe ‘Vorgehensweise bei Fehlern ’.
Siehe auch PCube_moveRampInc, PCube_moveStepInc, PCube_moveCurrentInc

8.36. PCube_moveVelExtended

Aufgabe Bewegungskommando im Strommodus mit erweiterter Statusrückgabe. Verfügbar ab PowerCube™-Betriebssystemversion 2.5.16 bzw. 3.5.14

Prototyp
```
int PCube_moveVelExtended( int dev, int cube, float vel, unsigned
long *stateRet, unsigned char *dioRet,  float *posRet);
```

Beschreibung Die Funktion wird wie PCube_moveVel mit dem Unterschied eingesetzt , daß über die Zeiger *stateRet*, *dioRet* und *posRet* ein Kurzstatus, der Zustand der internen digitalen IOs und die aktuelle Position des Moduls zurückgeliefert werden und für eine sofortige Auswertung zur Verfügung stehen.

R¸ckgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 82

Bei anderen Rückgabewerten siehe ‘Vorgehensweise bei Fehlern ’.
Siehe auch PCube_moveRampExtended, PCube_moveStepExtended, PCube_moveCurrentExtended

8.37. PCube_moveCurrent

Aufgabe Bewegungskommando im Strommodus

Prototyp
```
int PCube_moveCurrent( int dev, int cube, float cur);
```

Beschreibung Das Antriebsmodul *cube* wird in den Strom-Modus überführt.
Das Modul bewegt sich kontinuierlich mit der angegebenen Stromstärke, bis ein anderes Bewegungskommando oder ein HALT-Kommando gesendet wird oder bis die Softwareendlagen erreicht werden.

Dieses Kommando ist insbesondere dann zu verwenden, wenn das Modul durch Fremdeinwirkung oder durch falsche Bewegungskommandos über eine Hardwareendlage hinausgefahren wurde. Eine Möglichkeit, das Modul manuell aus dieser Position zu bewegen, ist der Aufruf von PCube_moveCurrent mit dem Parameter 0. Damit wird die Stromzuführung zum Antrieb unterbunden und eine eventuell vorhandene Magnetbremse gelöst. Andererseits ist es auch möglich, durch Angabe kleiner Stromstärken (1 bis 2 Ampere) softwaregesteuert aus der Endlagenposition herauszufahren (Richtung der Bewegung wird mit Vorzeichen des Stromstärkeparameters beeinflußt). Die Wahl größerer Stromstärken kann bei falscher Bewegungsrichtung das Modul beschädigen!

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 83

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_move...

8.38. PCube_moveCurrentInc

Aufgabe Bewegungskommando im Strommodus

Prototyp

```
int PCube_moveCurrentInc( int dev, int cube, Int16 cur);
```

Beschreibung Die Funktion wird wie PCube_moveCurrent mit dem Unterschied eingesetzt , daß hier Integer-Werte statt Float-Parameter übergeben werden können. Zur Dimensionierung der Parameter siehe unter Maßeinheiten der PowerCube™-Module.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 84

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_moveRampInc, PCube_moveStepInc, PCube_moveVelInc

8.39. PCube_moveCurrentExtended

Aufgabe Bewegungskommando im Strommodus mit erweiterter Statusrückgabe. Verfügbar ab PowerCube™-Betriebssystemversion 2.5.16 bzw. 3.5.13

Prototyp

```
int PCube_moveCurrentExtended( int dev, int cube, float cur,
unsigned long *stateRet, unsigned char *dioRet,  float *posRet);
```

Beschreibung Die Funktion wird wie PCube_moveCurrent mit dem Unterschied eingesetzt , daß über die Zeiger stateRet, dioRet und posRet ein Kurzstatus, der Zustand der internen digitalen IOs und die aktuelle Position des Moduls zurückgeliefert werden und für eine sofortige Auswertung zur Verfügung stehen.

Rückgabewert

CLD_OK	Bewegung wurde ausgelöst.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 85

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_moveRampExtended, PCube_moveStepExtended, PCube_moveVelExtended

8.40. Abfragen des aktuellen Modulzustands

8.41. PCube_getActPos

Aufgabe Holt die aktuelle Position des Moduls

Prototyp
```
int PCube_getActPos( int dev, int cube, float* pValue );
```

Beschreibung Die aktuelle Position des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Der zulässige Bereich der Modul erreichbaren Positionen ist mit Hilfe der Funktion PCube_setMinPos und PCube_setMaxPos programmierbar.
Die Einheit der Position ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Aktuelle Position wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 86

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getActVel, PCube_getDeltaPos

8.42. PCube_getActVel

Aufgabe Holt die aktuelle Geschwindigkeit des Moduls

Prototyp
```
PCube_getActVel( int dev, int cube, float* pValue );
```

Beschreibung Die aktuelle Geschwindigkeit des Antriebsmoduls cube wird über den Zeiger pValue übergeben. Dies ist die aus den Vorgaben der Bewegungssteuerung resultierende Geschwindigkeit des Moduls. Die zulässige Maximal-Geschwindigkeit des Moduls ist mit Hilfe der Funktion PCube_setMaxVel programmierbar.
Die Einheit der Geschwindigkeit ist je nach Modultyp m/s oder rad/s.

Rückgabewert

CLD_OK	Aktuelle Geschwindigkeit wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation

Tabelle 87

CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 87

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getActPos

8.43. PCube_getDeltaPos

Aufgabe Holt den aktuellen Schleppfehler des Moduls

Prototyp

```
int PCube_getDeltaPos( int dev, int cube, float* pValue );
```

Beschreibung Der aktuelle Schleppfehler des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Der Schleppfehler ist die Differenz zwischen der von der Bewegungssteuerung berechneten Soll-Position und der vom Geber des Moduls ermittelten Ist-Position. Der zulässige maximale Schleppfehler des Moduls ist mit Hilfe der Funktion PCube_setMaxDeltaPos programmierbar.

Rückgabewert

CLD_OK	Aktueller Schleppfehler wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 88

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getActPos, PCube_setMaxDeltaPos

8.44. PCube_getCur

Aufgabe Holt die aktuelle Stromaufnahme der Endstufe des Moduls

Prototyp

```
int PCube_getCur( int dev, int cube, float* pValue );
```

Beschreibung Die aktuelle Stromaufnahme der Endstufe des Antriebsmoduls cube wird über den Zeiger pValue übergeben. Die zulässige maximale Stromaufnahme der Endstufe ist mit Hilfe der Funktion PCube_setMaxCur programmierbar.
Die Einheit der Stromaufnahme ist Ampere (A).

Rückgabewert

CLD_OK	Aktuelle Stromaufnahme wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 89

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMaxCur

8.45. Abfragen und Setzen von Begrenzungswerten

8.46. PCube_getMaxPos

Aufgabe Holt die maximale Position des Moduls

Prototyp
```
int PCube_getMaxPos( int dev, int cube, float* pValue );
```

Beschreibung Die maximale Position des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Die Einheit der Position ist je nach Modultyp m oder rad

Rückgabewert

CLD_OK	Maximale Position wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 90

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMaxPos

8.47. PCube_getMinPos

Aufgabe Holt die minimale Position des Moduls

Prototyp
```
int PCube_getMinPos( int dev, int cube, float* pValue );
```

Beschreibung Die minimale Position des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Die Einheit der Position ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Minimale Position wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 91

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMinPos

8.48. PCube_setMaxPos

Aufgabe Setzt die maximale Position des Moduls

Prototyp
```
int PCube_setMaxPos( int dev, int cube, float pValue );
```

Beschreibung Die maximale Position des Antriebsmoduls *cube* wird in *pValue* übergeben. Alle

weiteren Bewegungskommandos werden auf die angegebenen Maximalwerte begrenzt. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt.
Die Einheit der Position ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Maximale Position wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 92

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getMaxPos

8.49. PCube_setMinPos

Aufgabe Setzt die minimale Position des Moduls

Prototyp
```
int PCube_setMinPos( int dev, int cube, float pValue );
```

Beschreibung Die minimale Position des Antriebsmoduls *cube* wird in *pValue* übergeben. Alle weiteren Bewegungskommandos werden auf die angegebenen Minimalwerte begrenzt. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt
Die Einheit der Position ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Die minimale Position wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 93

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getMinPos

8.50. PCube_getHomeOffset

Aufgabe Holt die benutzerdefinierte Nullposition des Moduls.

Prototyp
```
int PCube_getHomeOffset( int dev, int cube, float *pValue );
```

Beschreibung Die benutzerdefinierte, zur Laufzeit änderbare Nullposition des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Nicht zu verwechseln mit PCube_getDefHomeOffset; holt die im Modul fest eingebrannte benutzerdefinierte Nullposition. Die Einheit der Position ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Benutzerdefinierte Nullposition wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 94

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setHomeOffset

8.51. PCube_setHomeOffset

Aufgabe Setzt die benutzerdefinierte Nullposition des Moduls.

Prototyp
```
int PCube_setHomeOffset( int dev, int cube, float pValue );
```

Beschreibung Die benutzerdefinierte Nullposition des Antriebsmoduls *cube* wird zur Laufzeit in *pValue* übergeben. Der neue Wert für das HomeOffset ist nur solange gültig, wie die Logik-Stromversorgung aufrecht erhalten wird. Die Einheit der Position ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Benutzerdefinierte Nullposition wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 95

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getHomeOffset

8.52. PCube_getMaxVel

Aufgabe Holt die maximale Geschwindigkeit des Moduls

Prototyp
```
int PCube_getMaxVel( int dev, int cube, float* pValue );
```

Beschreibung Die maximale Geschwindigkeit des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Die Einheit der Geschwindigkeit ist je nach Modultyp m/s oder rad/s

Rückgabewert

CLD_OK	Maximale Geschwindigkeit wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 96

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMaxVel

8.53. PCube_getMinVel

Aufgabe Holt die minimale Geschwindigkeit des Moduls

Prototyp
```
int PCube_getMinVel( int dev, int cube, float* pValue );
```

Beschreibung Die minimale Geschwindigkeit des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Die Einheit der Geschwindigkeit ist je nach Modultyp m/s oder rad/s.

Rückgabewert

CLD_OK	Minimale Geschwindigkeit wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 97

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMinPos

8.54. PCube_setMaxVel

Aufgabe Setzt die maximale Geschwindigkeit des Moduls

Prototyp
```
int PCube_setMaxVel ( int dev, int cube, float pValue );
```

Beschreibung Die maximale Geschwindigkeit des Antriebsmoduls *cube* wird in *pValue* übergeben. Alle weiteren Bewegungskommandos werden auf die angegebenen Maximalwerte begrenzt. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt.
Die Einheit der Geschwindigkeit ist je nach Modultyp m/s oder rad/s.

Rückgabewert

CLD_OK	Maximale Geschwindigkeit wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 98

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getMaxVel

8.55. PCube_setMinVel

Aufgabe Setzt die minimale Geschwindigkeit des Moduls

Prototyp
```
int PCube_setMinVel( int dev, int cube, float pValue );
```

Beschreibung Die minimale Geschwindigkeit des Antriebsmoduls *cube* wird in *pValue* übergeben.

Alle weiteren Bewegungskommandos werden auf die angegebenen Minimalwerte begrenzt. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt. Die Einheit der Geschwindigkeit ist je nach Modultyp m/s oder rad/s.

Rückgabewert

CLD_OK	Minimale Geschwindigkeit wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 99

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern* '.
Siehe auch PCube_getMinVel

8.56. PCube_getMaxAcc

Aufgabe Holt die maximale Beschleunigung des Moduls

Prototyp

```
int PCube_getMaxAcc( int dev, int cube, float* pValue );
```

Beschreibung Die maximale Beschleunigung des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben.

Die Einheit der Beschleunigung ist je nach Modultyp m/s^2 oder rad/s^2

Rückgabewert

CLD_OK	Maximale Beschleunigung wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 100

Bei anderen Rückgabewerten siehe 'Vorgehensweise bei Fehlern '.
Siehe auch PCube_setMaxAcc

8.57. PCube_setMaxAcc

Aufgabe Setzt die maximale Beschleunigung des Moduls

Prototyp int PCube_setMaxAcc(int dev, int cube, float pValue);

Beschreibung Die maximale Beschleunigung des Antriebsmoduls *cube* wird in *pValue* übergeben. Alle weiteren Bewegungskommandos werden auf die angegebenen Maximalwerte begrenzt. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt. Die Einheit der Beschleunigung ist je nach Modultyp m/s^2 oder rad/s^2.

Rückgabewert

CLD_OK	Maximale Beschleunigung wurde gesetzt.

Tabelle 101

CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 101

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getMaxAcc

8.58. PCube_getMinAcc

Aufgabe Holt die minimale Beschleunigung des Moduls

Prototyp

```
int PCube_getMinAcc( int dev, int cube, float pValue );
```

Beschreibung Die minimale Beschleunigung des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben.
Die Einheit der Beschleunigung ist je nach Modultyp m/s^2 oder rad/s^2.

Rückgabewert

CLD_OK	Minimale Beschleunigung wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 102

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMinAcc

8.59. PCube_setMinAcc

Aufgabe Setzt die minimale Beschleunigung des Moduls

Prototyp

```
int PCube_setMinAcc( int dev, int cube, float pValue );
```

Beschreibung Die minimale Beschleunigung des Antriebsmoduls *cube* wird in *pValue* übergeben. Alle weiteren Bewegungskommandos werden auf die angegebenen Maximalwerte begrenzt. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt. Die Einheit der Beschleunigung ist je nach Modultyp m/s^2 oder rad/s^2.

Rückgabewert

CLD_OK	Minimale Beschleunigung wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 103

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

Siehe auch PCube_getMinAcc

8.60. PCube_getMaxCur

Aufgabe Holt die maximal zulässige Stromaufnahme der Endstufe des Moduls

Prototyp
```
int PCube_getMaxCur( int dev, int cube, float* pValue );
```

Beschreibung Die maximal zulässige Stromaufnahme der Endstufe des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben.
Die Einheit der Stromaufnahme ist Ampere (A).

Rückgabewert

CLD_OK	Maximale Stromaufnahme wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 104

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMaxCur

8.61. PCube_setMaxCur

Aufgabe Setzt die maximal zulässige Stromaufnahme der Endstufe des Moduls

Prototyp
```
int PCube_setMaxCur( int dev, int cube, float* pValue );
```

Beschreibung Die maximal zulässige Stromaufnahme der Endstufe des Antriebsmoduls *cube* wird in *value* übergeben. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt.
Die Einheit der Stromaufnahme ist Ampere (A).

Rückgabewert

CLD_OK	Maximale Stromaufnahme wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 105

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getMaxCur

8.62. PCube_getMinCur

Aufgabe Holt die minimal zulässige Stromaufnahme der Endstufe des Moduls

Prototyp
```
int PCube_getMinCur( int dev, int cube, float* pValue );
```

Beschreibung Die minimal zulässige Stromaufnahme der Endstufe des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben.

Die Einheit der Stromaufnahme ist Ampere (A).

Rückgabewert

CLD_OK	Minimale Stromaufnahme wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 106

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMinCur

8.63. PCube_setMinCur

Aufgabe Setzt die minimal zulässige Stromaufnahme der Endstufe des Moduls

Prototyp

```
int PCube_setMinCur( int dev, int cube, float* pValue );
```

Beschreibung Die minimal zulässige Stromaufnahme der Endstufe des Antriebsmoduls *cube* wird in *value* übergeben. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt.
Die Einheit der Stromaufnahme ist Ampere (A).

Rückgabewert

CLD_OK	Minimale Stromaufnahme wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 107

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getMinCur

8.64. PCube_getMaxDeltaPos

Aufgabe Holt den maximal zulässigen Schleppfehler des Moduls

Prototyp

```
int PCube_getMaxDeltaPos( int dev, int cube, float* pValue );
```

Beschreibung Der maximal zulässige Schleppfehler des Antriebsmoduls *cube* wird über den Zeiger p*Value* übergeben.
Die Einheit des Schleppfehlers ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Maximaler Schleppfehler wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.

CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 108

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_setMaxDeltaPos

8.65. PCube_setMaxDeltaPos

Aufgabe Setzt den maximal zulässigen Schleppfehler des Moduls

Prototyp

```
int PCube_setMaxDeltaPos( int dev, int cube, float* pValue );
```

Beschreibung Der maximal zulässige Schleppfehler des Antriebsmoduls cube wird in value übergeben. Die angegebenen Werte dürfen die im Modul voreingestellten Werte nicht überschreiten. Höhere Werte werden auf die voreingestellten Werte begrenzt. Die Einheit des Schleppfehlers ist je nach Modultyp m oder rad.

Rückgabewert

CLD_OK	Maximaler Schleppfehler wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 109

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Siehe auch PCube_getMaxDeltaPos

8.66. Abfragen und Setzen der Koeffizienten des Lageregelkreises

Diese Funktionen stehen erst ab der Version 2.5.1 zur Verfügung

8.67. PCube_getC0

Aufgabe Holt den Koeffizienten C0 des Lageregelkreises

Prototyp

```
int PCube_getC0( int dev, int cube, Int16* pValue );
```

Beschreibung Der Koeffizient C0 des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Dieser Wert stellt einen Verstärkungsfaktor dar. Je größer C0 gesetzt ist, desto geringer ist die Verstärkung. Gültige Werte sind alle geraden Zahlen zwischen 12 und 48.

Rückgabewert

CLD_OK	Koeffizient C0 wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 110

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.68. PCube_getDamp

Aufgabe Holt den Dämpfungswert des Lageregelkreises

Prototyp
```
int PCube_getDamp( int dev, int cube, Int16* pValue );
```

Beschreibung Die Dämpfung des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Dieser Wert stellt einen Dämpfungsfaktor dar. Je größer Damp gesetzt ist, desto größer ist die Dämpfung. Gültige Werte liegen zwischen 1 und 4.

Rückgabewert

CLD_OK	Dämpfungswert Damp wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 111

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.69. PCube_getA0

Aufgabe Holt den Koeffizienten A0 des Lageregelkreises

Prototyp
```
int PCube_getA0( int dev, int cube, Int16* pValue );
```

Beschreibung Der Koeffizient A0 des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Hier wird die Beschleunigungsfähigkeit des Antriebs angezeigt. Je größer A0 gesetzt ist, desto größere Beschleunigungsfähigkeit wird unterstellt. Gültige Werte liegen zwischen 1 und 12.

Rückgabewert

CLD_OK	Koeffizient A0 wurde geholt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 112

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.70. PCube_setC0

Aufgabe Setzt den Koeffizienten C0 des Lageregelkreises

Prototyp
```
int PCube_setC0( int dev, int cube, Int16 pValue );
```

Beschreibung Der Koeffizient C0 des Antriebsmoduls *cube* wird in *value* übergeben. Dieser Wert

stellt einen Verstärkungsfaktor dar. Je größer C0 gesetzt wird, desto geringer wird die Verstärkung. Gültige Werte sind alle geraden Zahlen zwischen 12 und 48. Nach Setzen des oder der Koeffizienten muß PCube_recalcPIDParams aufgerufen werden, damit die Änderungen wirksam werden.

Rückgabewert

CLD_OK	Koeffizient C0 wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 113

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.71. PCube_setDamp

Aufgabe Setzt den Dämpfungswert des Lageregelkreises

Prototyp

```
int PCube_setDamp( int dev, int cube, Int16 pValue );
```

Beschreibung Die Dämpfung des Antriebsmoduls *cube* wird in *value* übergeben. Dieser Wert stellt einen Dämpfungsfaktor dar. Je größer Damp gesetzt wird, desto größer wird die Dämpfung. Gültige Werte liegen zwischen 1 und 4. Nach Setzen des oder der Koeffizienten muß PCube_recalcPIDParams aufgerufen werden, damit die Änderungen wirksam werden.

Rückgabewert

CLD_OK	Dämpfungswert Damp wurde gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 114

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.72. PCube_setA0

Aufgabe Setzt den Koeffizienten A0 des Lageregelkreises

Prototyp

```
int PCube_setA0( int dev, int cube, Int16 pValue );
```

Beschreibung Der Koeffizient A0 des Antriebsmoduls *cube* wird über den Zeiger *pValue* übergeben. Hier wird die Beschleunigungsfähigkeit des Antriebs angesetzt. Je größer A0 gesetzt wird, desto größere Beschleunigungsfähigkeit wird unterstellt. Gültige Werte liegen zwischen 1 und 12. Nach Setzen des oder der Koeffizienten muß PCube_recalcPIDParams aufgerufen werden, damit die Änderungen wirksam werden.

Rückgabewert

CLD_OK	Koeffizient A0 wurde gesetzt.

Tabelle 115

CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 115

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.73. PCube_recalcPIDParams

Aufgabe Neuberechnen des Lageregelkreises

Prototyp
```
int PCube_recalcPIDParams( int dev, int cube );
```

Beschreibung Die Neuberechnung des Lageregelkreises des Antriebsmoduls cube wird durchgeführt. Nach Setzen der Reglerparameter mittels PCube_setC0, PCube_setDamp oder PCube_setA0 muß diese Funktion aufgerufen werden, damit die Änderungen wirksam werden.

Rückgabewert

CLD_OK	Neuberechnung der Regelschleife wird durchgeführt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 116

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.74. Abfragen und Setzen der internen Digital-Ein- und Ausgänge

8.75. PCube_getDioData

Aufgabe Holt die aktuell gesetzten Bits der internen Digital-Ein- und Ausgänge. Das Bit-Register wird über die Define-Konstante ACT_DIOSETUP angesprochen. Dies ist ein 32-Bit-Wort, das in der Variablen, auf die der Zeiger *pValue* zeigt, abgelegt wird. Nur die ersten 8 Bits (ab Betriebssystemversion 2.5.16 bzw. 3.5.13 die ersten 11 Bits) in diesem Register haben eine bestimmte Bedeutung. Die Bits 1 - 4 repräsentieren die Inbits (Eingänge), die Bits 5 - 8 repräsentieren die Outbits (Ausgänge). Ab Cube-Version 2.5.16 bzw. 3.5.13 repräsentieren die Bits 9 - 11 den Zustand der End- bzw. Referenz-schalter (siehe Bedeutung der Bits im 32-Bit Wort DioSetup).

Prototyp
```
PCube_getDioData(int dev, int cube, UInt32 *bits)
```

Beschreibung Die aktuell gesetzten Bits der Ein- und Ausgänge der internen DIOs des Moduls *cube* werden über den Zeiger *bits* übergeben.

Rückgabewert

CLD_OK	Bitregister der modulinternen DIOs wurde abgefordert.

Tabelle 117

CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 117

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.

8.76. PCube_setDioData

Aufgabe Setzt die Bits der internen Digital-Ein- und Ausgänge.

Prototyp `PCube_setDioData(int dev, int cube, UInt32 bits)`

Beschreibung Die Bits der Ein- und Ausgänge der internen DIOs des Moduls *cube* werden in *bits* übergeben.

Rückgabewert

CLD_OK	Bitregister der modulinternen DIOs wurde neu gesetzt.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 118

Bei anderen Rückgabewerten siehe '*Vorgehensweise bei Fehlern*'.
Beispiel für das Lesen und Setzen der Bits interner DIOs:

```
int ival=0;
UInt32 ivalold;
char buffer[33];
retVal = PCube_getDioData( dev,idMap[module],&ivalold);
printf( "... Set Bit Nr (0-7): " );
cscanf( "%d", &ival);
ivalold=ivalold|(UInt32)pow(2,ival);
retVal = PCube_setDioData( dev,idMap[module],ivalold);
retVal = PCube_getDioData( dev,idMap[module],&ivalold);
printf("\nNew Settings:%s\n",_itoa( ivalold, buffer, 2 ));
```

8.77. Abfrage von Vorgabewerten und sonstigen Funktionen

Die im vorhergehenden Abschnitt dargestellten Funktionen dienen zum Abfragen und Setzen von Vorgabewerten in den Antriebsmodulen, deren Überschreitung in der Regel einen Fehlerzustand bewirkt. Wie beschrieben, geht in diesem Fall geht das Modul in den Sicherheitszustand über. Die im folgenden genannten Funktionen PCube_getDefXXX dienen zum Abfragen der im Modul voreingestellten Maximalwerte, die nicht überschritten werden dürfen.
Wenn in den im vorhergehenden Abschnitt gezeigten Funktionen höhere als die voreingestellten Werte programmiert werden sollen, werden die neuen Werte in jedem Fall auf die voreingestellten begrenzt.
Auf eine ausfühliche Beschreibung der Funktionen wird hier verzichtet, da die Parameter denen der im vorherigen Abschnitt beschriebenen Funktionen entsprechen.
Folgende Funktionen stehen zur Verfügung:

- `int PCube_getDefHomeOffset( int dev, int cube,`

```
float* pValue );
```

- `int PCube_getDefGearRatio( int dev, int cube, float* pValue );`
- `int PCube_getDefLinearRatio( int dev, int cube, float* pValue );`
- `int PCube_getDefMinPos( int dev, int cube, float* pValue );`
- `int PCube_getDefMaxPos( int dev, int cube, float* pValue );`
- `int PCube_getDefMaxDeltaPos( int dev, int cube, float* pValue );`
- `int PCube_getDefMaxDeltaVel( int dev, int cube, float* pValue );`
- `int PCube_getDefTorqueRatio( int dev, int cube, float* pValue );`
- `int PCube_getDefCurRatio( int dev, int cube, float* pValue );`
- `int PCube_getDefMinVel( int dev, int cube, float* pValue );`
- `int PCube_getDefMaxVel( int dev, int cube, float* pValue );`
- `int PCube_getDefMinAcc( int dev, int cube, float* pValue );`
- `int PCube_getDefMaxAcc( int dev, int cube, float* pValue );`
- `int PCube_getDefMinCur( int dev, int cube, float* pValue );`
- `int PCube_getDefMaxCur( int dev, int cube, float* pValue );`
- `int PCube_getDefHomeVel( int dev, int cube, float* pValue );`
- `int PCube_getDefHomeAcc( int dev, int cube, float* pValue );`
- `int PCube_getDefKpPosControl( int dev, int cube, float* pValue );`
- `int PCube_getDefTnPosControl( int dev, int cube, float* pValue );`
- `int PCube_getDefTvPosControl( int dev, int cube, float* pValue );`
- `int PCube_getDefT1PosControl( int dev, int cube, float* pValue );`
- `int PCube_getDefKpVelControl( int dev, int cube, float* pValue );`
- `int PCube_getDefTnVelControl( int dev, int cube, float* pValue );`
- `int PCube_getDefTvVelControl( int dev, int cube, float* pValue );`
- `int PCube_getDefT1VelControl( int dev, int cube, float* pValue );`
- `int PCube_getDefDioSetup( int dev, int cube, UInt32* pValue );`
- `int PCube_getDefSerialNo( int dev, int cube, UInt32* pValue );`
- `int PCube_getDefConfig( int dev, int cube, UInt32* pValue );`
- `int PCube_getDefPulsesPerTurn( int dev, int cube, UInt32* pValue );`
- `int PCube_getDefCubeVersion( int dev, int cube, UInt16* pValue );`
- `int PCube_getDefServiceInterval( int dev, int cube, UInt16* pValue );`
- `int PCube_getDefBrakeTimeOut( int dev, int cube, UInt16* pValue );`

- `int PCube_getDefAddress( int dev, int cube, UInt8* pValue );`
- `int PCube_getDefMoveMode( int dev, int cube, UInt8* pValue );`
- `int PCube_getDefPrimaryBaudrate( int dev, int cube, UInt8* pValue );`
- `int PCube_getDefSecondaryBaudrate( int dev, int cube, UInt8* pValue );`
- `int PCube_getDefC0( int dev, int cube, Int16* pValue );`
- `int PCube_getDefDamp( int dev, int cube, Int16* pValue );`
- `int PCube_getDefA0( int dev, int cube, UInt16* pValue );`

8.78. Die I/O-Schnittstellen

Die in der m5dll enthaltenen Funktionen zum Betreiben von I/O-Modulen der Firma EMS Thomas Wünsche werden im folgenden beschrieben. Es werden zur Zeit zwei Modultypen unterstützt: 0=CST-DO8-24V und 1=CST-DI8-24V.

8.79. TW_initModules

Aufgabe

Diese Funktion initialisiert ein Modul und kann vor Benutzung der weiteren Kommandos aufgerufen werden. Es ist zu beachten, daß diese Funktion nur verwendet werden darf, wenn sich lediglich ein TW-Modul am Bus befindet. Die Synchronisation mehrerer TW-Module wird durch Senden von 100 mit Zufallswerten initialisierten CAN-Telegrammen bei Aufruf der Funktion PCube_openDevice vorgenommen. Beim anschließenden Aufruf der Funktion TW_initModules geht diese Synchronisation wieder verloren.

Prototyp

```
int TW_initModules( int device );
```

Beschreibung

Diese Funktion ist nach Spannungsabfall in der Energieversorgung aufzurufen. Das Modul wird innerhalb dieser Funktion auf die eingestellte Baudrate vorbereitet.
Der Parameter *device* bezeichnet die Schnittstelle, über die das Modul angesteuert wird.

R¸ckgabewert

CLD_OK	Modul wurde erkannt.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.
Anderer Wert	Es wurde kein Modul am Bus gefunden.

Tabelle 119

8.80. TW_addModule

Aufgabe

Diese Funktion fügt ein Modul bestimmten Typs mit definierter Seriennummer der Laufzeitumgebung hinzu. Nach erfolgreichem Hinzufügen kann auf die I/O-Ports dieses Moduls zugegriffen werden.

Prototyp

```
int TW_addModule( int device, UInt8 typ, UInt32 serialnumber, int
```

```
*index );
```

Beschreibung

Der Parameter *device* bezeichnet die Schnittstelle, worüber die Module angesteuert werden. Die anzugebende Seriennummer ist diejenige, die auf dem Modul aufgedruckt ist. Die Werte für den Parameter *typ* können Sie der m5.h-Datei entnehmen:

Typ	Konstante	Wert
CST-Digital-Ausgangsmodul mit 8 Ausgängen 24V	TW_DIG24V_OUT	0
CST-Digital-Eingangsmodul mit 8 Eingängen 24V	TW_DIG24V_IN	1
CST-Analog-Ausgangsmodul mit 2 Ausgängen 0-10V/12Bit	TW_ANA10V_OUT	2
CST-Analog-Eingangsmodul mit 4 Eingängen 0-10V/ 12Bit	TW_ANA10V_IN	3
CST-Analog-Ausgangsmodul mit 2 Ausgängen 0-25mA/12Bit	TW_ANA25MA_OUT	4
CST-Analog-Eingangsmodul mit 4 Eingängen 0-25mA/12Bit	TW_ANA25MA_IN	5

Tabelle 120

Rückgabewert

CLD_OK	Module wurden erkannt.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.
Anderer Wert	Modul wurde nicht korrekt hinzugefügt.

Tabelle 121

8.81. TW_getIO

Aufgabe

Diese Funktion fragt den Zustand eines I/O-Ports ab.

Prototyp

```
int TW_getIO( int device, int index, int port, Float *state );
```

Beschreibung

Der Parameter *device* bezeichnet die Schnittstelle, worüber die Module angesteuert werden. Die angegebene Seriennummer bestimmt das Modul, das zuvor mit TW_addModule() hinzugefügt wurde.
Der Parameter *port* selektiert den Eingangs- oder Ausgangsport (0-7 bei digtalen I/O's, 0-1/4 bei analogen I/O's), dessen Zustand gelesen werden soll. Der Zustand des Moduls kann unabhängig vom Typ des Moduls gelesen werden.
In der Variablen *state* wird anschließend der Zustand des Ports geliefert, wobei 0 nicht aktiv und 1 aktiv ist (bei digitalen I/O's). Analoge Module liefern den abgetasteten Spannungswert (0-10V) bzw. den abgetasteten Stromwert (0-25mA).

Rückgabewert

CLD_OK	Zustand wurde gelesen.
CLDERR_BADPARAM	Zustand wurde nicht gesetzt, da falsche Parameter übergeben wurden.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.

CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 122

8.82. TW_setIO

Aufgabe Diese Funktion setzt den Zustand eines I/O-Ports.

Prototyp

```
int TW_setIO( int device, int index, int port, Float state );
```

Beschreibung Der Parameter *device* bezeichnet die Schnittstelle, worüber die Module angesteuert werden. Der angegebene Index bestimmt das Modul, das zuvor mit TW_addModule() hinzugefügt wurde. Der Parameter *port* selektiert den Eingangs- oder Ausgangsport (0-7 bei digtalen I/O's, 0-1 bei analogen I/O's), dessen Zustand gesetzt werden soll. Der Zustand des Moduls kann unabhängig vom Typ des Moduls gesetzt werden. Allerdings erfolgt nur bei Output-Modulen wirklich eine Änderung des Schaltvorgangs. In der Variablen *state* wird anschließend der Zustand des Ports gesetzt, wobei 0 nicht aktiv und 1 aktiv ist (bei digitalen I/O's). Analoge Module werden mit einem Spannungswert (0-10V) bzw. einen Stromwert (0-25mA) bedient.

Rückgabewert

CLD_OK	Zustand wurde gesetzt.
CLDERR_BADPARAM	Zustand wurde nicht gesetzt, da falsche Parameter übergeben wurden.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 123

8.83. TW_getDigIOall

Aufgabe Diese Funktion liest den Zustand aller Channels eines digitalen TW-Moduls.

Prototyp

```
int TW_getDigIOall( int device, int index, UInt8 *value );
```

Beschreibung Der Parameter *device* bezeichnet die Schnittstelle, worüber die Module angesteuert werden. Die angegebene Seriennummer bestimmt das Modul, das zuvor mit TW_addModule() hinzugefügt wurde.
In der Variablen *value* wird der Zustand der Channels geliefert, wobei 0 nicht aktiv und 1 aktiv ist.

Rückgabewert

CLD_OK	Zustand wurde gelesen.
CLDERR_BADPARAM	Zustand wurde nicht gesetzt, da falsche Parameter übergeben wurden.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 124

8.84. TW_setDigIOall

Aufgabe Diese Funktion setzt alle Channels eines digitalen TW-Moduls.

Prototyp

```
int TW_setDigIOall( int device, int index, UInt8 value );
```

Beschreibung Der Parameter *device* bezeichnet die Schnittstelle, worüber die Module angesteuert werden. Die angegebene Seriennummer bestimmt das Modul, das zuvor mit TW_addModule() hinzugefügt wurde. Über die Variable *value* wird der Zustand aller Channels gesetzt, wobei 0 nicht aktiv und 1 aktiv ist.

Rückgabewert

CLD_OK	Zustand wurde gelesen.
CLDERR_BADPARAM	Zustand wurde nicht gesetzt, da falsche Parameter übergeben wurden.
CLDERR_TRANSMISSIONERROR	Übertragungsfehler bei der Kommunikation
CLDERR_DRIVE_MODULETIMEOUT	Antriebsmodul hat nicht reagiert.
CLERR_WRONGHANDLE	Verwendeter Zeiger auf Schnittstelle nicht initialisiert oder bereits gelöscht.

Tabelle 125

8.85. Besonderheiten bei speziellen Implementierungen

OR-PC5 CAN Diese Schnittstelle kann zusammen mit dem unixbasierten Echtzeitbetriebssystem QNX verwendet werden. In der Implementierung des CAN-Treibers für die or Industrial Computers PC-5 Serie sind einige Besonderheiten zu beachten. So besteht der Treiber aus zwei Teilen:

- Den CAN-Server, der die Kommunikation mit dem CAN-Bus übernimmt.
- Dem Softwaretreiber, der als lib vorliegt und zur Applikation hinzugelinkt werden muß. Dieser übernimmt die Kommunikation mit dem CAN-Server.

Der CAN-Serverprozeß `m5candrv` muß gestartet sein, bevor ein Client gestartet werden soll. Dies ist sichergestellt, da der Clientprozeß anderenfalls eine Fehlermeldung produziert. Des weiteren muß der Queuehandler von QNX gestartet sein. Dies kann in einem Startupscript automatisiert werden. Der Handler lautet *Mqueue* und ist mittels `Mqueue &` zu starten.
Im Lieferumfang der Treiberdistribution befindet sich ein Beispielprogramm im Sourcecode und ein dazugehöriges Makefile, das eine lauffähige Version des Programms *m5apitst* erzeugt.
Dieses ist mit folgender Kommandozeile zu starten: `./m5apitst ORCAN:1`
Anschließend wird ein etwa 30 s andauernder Scan der angeschlossenen Module durchgeführt. Dies sollten PowerCube™-M5-Module sein, die möglichst nicht im M3-Modus betrieben werden.
Die Treiber sind mit root-Berechtigung zu starten.

8.86. Beispiel-Implementierungen in unterschiedliche Arbeitsumgebungen

MS Visual C++ Um die Funktionstüchtigkeit und Bewegungsmodi der PowerCube™-Module testen zu können, wurde das Programm *PCubeDemo* geschrieben. Das Testprogramm wurde in einer MS Visual C ++ Umgebung, Version 6 als dialogbasierte MFC-Anwendung entworfen. Im Folgenden werden die Projektplanung und -erstellung ausführlich beschrieben: Das Testprogramm PCubeDemo.

9. BAPS und PowerCube™

Diese Anleitung erläutert den Betrieb der PowerCube™-Module an einer Rho4-Robotersteuerung für die Scara-Typen SR6/SR8 von Bosch. Es werden die erforderlichen Hardwarevoraussetzungen, Einstellungen sowie ein Testprogramm in der Programmiersprache BAPS erklärt.
PowerCube™-Module eignen sich in diesem Zusammenhang besonders zur Ergänzung der SR6/SR8-Scara-Roboter durch eine fünfte und sechste Achse (PW070) sowie durch elektrische Greifer (PG070) oder weitere periphere Positionierachsen aus der PowerCube™-Produktlinie.

9.1. Hardwarevoraussetzungen

Der Anschluß von PowerCube™-Modulen an den Steuerschrank IQ200 (Bosch) erfolgt anhand der seriellen V24-Schnittstelle der Rho4.
Die Rho4.1 verfügt über 4 freie V24 (RS232)-Schnittstellen, bezeichnet durch:

Software	Hardware	Anschluß
V24_1	SER_1	X31
V24_2	SER_2	X32
V24_3	SER_3	X33
V24_4	SER_4	X34

Tabelle 126

Die einzusetzenden PowerCube™-Module müssen dementsprechend für serielle RS232-Kommunikation konfiguriert sein. Die Rho4.1 unterstützt Baudraten bis 38400 bit/s. Da die PowerCube™-Module standardmäßig auf 9600 bit/s eingestellt sind, empfiehlt es sich zunächst, diese Defaulteinstellung beizubehalten. Die an der Rho4.1 erforderlichen Einstellungen werden im folgenden Abschnitt (Einstellungen) erläutert.

Kabelbelegung

Die PowerCube™ RS232-Masterkabel aus dem Standard-Lieferprogramm sind zum direkten Anschluß an die IQ200 geeignet. Das Pinning ist folgenderweise:

Pin	IQ200	PowerCube™-System
1	DTR 20mA	-
2	RXD	TXD
3	TXD	-
4	DTR	SGND
5	SGND	RXD
6	DSR	-
7	RTS/RXD 20mA	-
8	Frei	-
9	TXD 20mA	-
24V	-	A1
0V	-	A2

Tabelle 127

9.2. Einstellungen

PowerCube™-Module mit RS232-Anschluß haben standardseitig folgende Einstellung:

9600 bit/s, 8 bit Daten, 1 Stopbit, keine Parität, kein Handshake

Um PowerCube™-Module an der Rho4.1 betreiben zu können, müssen die Default-Einstellungen der ausgewählten V24 (SER_1,.., SER_4) geändert werden. Dazu wird das "Menü Geräte-Datei-EA/Voreinstellungen" am PHG aufgerufen (dies ist auch anhand des Virtual PHG unter Windows möglich). Die erforderlichen Einstellungen werden exemplarisch am Beispiel der SER_3 (Unterpunkt 5) demonstriert:

- Serieller Kanal = 0
- Protokoll = 5
- Schnittstelle = 3
- Baudrate = 9600
- Stopbit Anzahl = 1.0
- Parität = keine
- Wortlänge = 8
- SW/HW-Handshake = 0
- Timeout beim Lesen = 1000
- Timeout beim Schreiben = 1000

Zum Abspeichern dieser Einstellungen muß der Notaus am PHG oder an der IQ200 betätigt sein. Damit sind die neuen Default-Einstellungen dauerhaft abgelegt.

9.3. Software

Für den Betrieb der PowerCube™-Module an der Rho4.1 im Zusammenhang mit der Bewegung der Standard-Roboterachsen eignet sich die Bosch-Programmiersprache BAPS. Sie gestattet die gleichzeitige Ansteuerung des SR6/SR8 und der PowerCube™-Achsen.
Dieser Dokumentation ist ein Beispielprogramm zur Veranschaulichung der Programmierung beigefügt (PCTest2.qll).
Um das Testprogramm PCTest2 betreiben zu können, werden folgende Dateien benötigt:

Quelldatei (auf PC)	Ausführbare Datei (auf Rho4)	Inhalt/Bedeutung
PCTest2.qll	PCTest2.ird	Testprogramm zur Ansteuerung von SR8 und PW070
Pcube.inc	-	Kapselung der PowerCube™-Funktionen als Unterprogramme in einer separaten Include-Datei
PCConst.inc	-	Konstantenvereinbarung für PowerCube™-Funktionen, gekapselt in einer separaten Include-Datei
Exprog.dat	Exprog.dat	Feste Datei für Rho4, die das auszuführende Programm angibt
Init.qll	Init.ird	Standard-Referenzfahrt-Programm für SR8

Tabelle 128

9.4. Erzeugen der ausführbaren Dateien und Übertragung auf die Rho4

Bevor begonnen werden kann, sind die Dateien PCTest2.qll, Pcube.inc, PCConst.inc,

exprog.dat und init.qll vom mitgelieferten Datenträger in ein separates Verzeichnis des Rho4-PC zu kopieren (die Read-Only-Markierung muß nach dem Kopieren von CD gelöscht werden).
Die ausführbaren IRD-Dateien entstehen durch Aufruf des BAPS-Übersetzers unter Windows. Das Programm Rhops4/online ist zu starten und unter "Einstellungen" der Editor aufzurufen. Im Editor ist die Datei PCTest.qll zu öffnen und Werkzeug/BAPS Übersetzer auszuwählen. Das Programm wird nun kompiliert. Es erfolgt eine Meldung über den Erfolg. In Rhops4/online ist nun die Liste der PC-Dateien (F2) zu öffnen. Dort findet man die Datei PCTest2.ird, die durch den Übersetzer neu erstellt wurde. Die Datei PCTest2.ird und exprog.dat ist zu markieren und an die Rho4 mit F5 zu übertragen (Betätigen des Button "Überschreiben").
Im Editor ist init.qll zu öffnen und der Übersetzer erneut aufzurufen. Der Vorgang "Datei übertragen" mit der nun entstandenen Datei Init.ird ist zu wiederholen.
Dieses Programm ist zur Referenzfahrt des Scara-Roboters notwendig und muß sich auf der Rho4 befinden.

9.5. Starten der Programme

Nach Übertragung der ausführbaren Dateien auf die Rho4 können die Programme durch Betätigen der Taste PROG an der IQ200 aufgerufen werden.
Zunächst sind die Antriebsverstärker zuzuschalten und PROG betätigt. Wenn der Scara noch nicht referenziert war, wird zunächst automatisch das Programm "Init" ausgeführt (Referenzfahrt). Anderenfalls wird das Programm PCTest ausgeführt (wie in exprog.dat festgelegt).
Vor dem ersten Start des Programms PCTest sollte der folgende Abschnitt gelesen werden. Er enthält, welche Funktionen darin ausgeführt werden.

9.6. Programmieren mit den PowerCube™-Funktionen

Folgende Funktionen wurden in BAPS als Unterprogramme realisiert:

Funktionsname	Parameter	Verwendungszweck
PCGetPos	Moduladresse, Variable für Position	Abfrage der aktuellen Position in Inkrementen
PCSetRampVel	Moduladresse, Rampengeschwindigkeit	Setzt die für PCMoveRamp gültige Geschwindigkeit in Inkrementen/s
PCSetRampAcc	Moduladresse, Rampenbeschleunigung	Setzt die für PCMoveRamp gültige Beschleunigung in Inkrementen/s²
PCSetConfig	Moduladresse, Konfigurationswort	Setzt das Konfigurationswort
PCGetConfig	Moduladresse, Variable für Konfigurationswort	Liest das Konfigurationswort
PCSetMaxPos	Moduladresse, Maximalposition	Setzt die Software-Endlage (Maximalposition) in Inkrementen
PCGetMaxPos	Moduladresse, Variable für Maximalposition	Liest die Software-Endlage (Maximalposition) in Inkrementen
PCSetMinPos	Moduladresse, Minimalposition	Setzt die Software-Endlage (Minimalposition) in Inkrementen
PCGetMinPos	Moduladresse, Variable für Minimalposition	Liest die Software-Endlage (Minimalposition) in Inkrementen

Tabelle 129

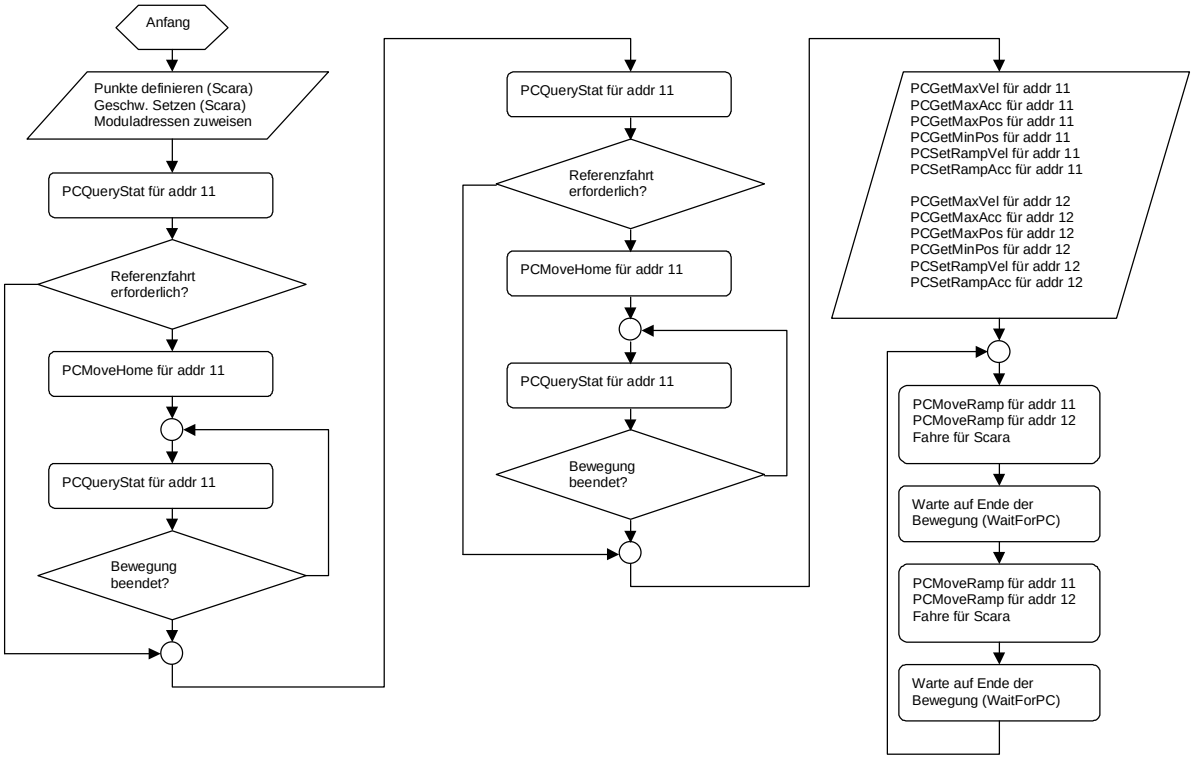
PCSetMaxVel	Moduladresse, Maximalgeschwindigkeit	Setzt die maximal zulässige Geschwindigkeit in Inkrementen/s
PCGetMaxVel	Moduladresse, Variable für Maximalgeschwindigkeit	Liest die maximal zulässige Geschwindigkeit in Inkrementen/s
PCSetMaxAcc	Moduladresse, Maximalbeschleunigung	Setzt die maximal zulässige Beschleunigung in Inkrementen/s²
PCGetMaxAcc	Moduladresse, Variable für Maximalbeschleunigung	Liest die maximal zulässige Beschleunigung in Inkrementen/s²
PCMoveHome	Moduladresse	Startet eine Referenzfahrt
PCHalt	Moduladresse	Stoppt die laufende Bewegung und versetzt den Antrieb in den HALT-Modus
PCReset	Moduladresse	Löscht anstehende Fehlermeldungen sowie den HALT-Modus
PCMoveRamp	Moduladresse, Zielposition	Startet eine rampenförmige Positionsfahrt (Vorgabe in Inkrementen). Das Ende der Reise wird anhand der Ende-Meldung bei PCQueryStat angezeigt.
PCMoveVel	Moduladresse, Geschwindigkeit	Startet eine Fahrt mit konstanter Geschwindigkeit (Vorgabe in Inkrementen/s)
PCMoveCur	Moduladresse, Strom	Startet eine Fahrt mit konstantem Strom (Vorgabe in digits). Wertebereich ± 500. Dabei entsprechen 500 digits einem Strom von 15A bzw. 30A (entsprechend der Angabe "Peak Current" im Produktbegleitbrief)
PCGetPID	Moduladresse, Variablen für C0, D und A0	Abfrage der Reglerparameter C0, Dämpfung und A0
PCSetPID	Moduladresse, C0, D, A0	Setzt die Reglerparameter C0, Dämpfung und A0
PCQueryStat	Moduladresse, Variablen für Fehler, Bewegung, Ende, Referenz, Halt	Diese Funktion fragt den aktuellen Status ab (wie PCGetStatus) und wertet die wichtigsten Statusflags aus. Angezeigt werden ein anstehender Fehlerzustand, eine laufende Bewegung, das Ende einer Rampenfahrt, den Referenzzustand der Achse sowie der HALT-Modus. Mit Hilfe von PCReset kann der HALT-Modus gelöscht werden.
PCGetStatus	Moduladresse, Variable für Statuswort	Diese Funktion dient der Abfrage des aktuellen Modulstatus. Sie kann eingesetzt werden, z.B. um den Grund für einen anstehenden Fehler zu erfahren

Tabelle 129

Jedes PowerCube™-Modul wird eindeutig anhand seiner Adresse am Bus identifiziert. Die Moduladresse ist dem Produktbegleitbrief zu entnehmen (die Adresse kann Werte von 1 bis 31 annehmen).
Jede Funktion der oben genannten Funktionen hat zudem einen weiteren Parameter, der als Rückgabecode für den Erfolg eines Funktionsaufrufs interpretiert wird. Ist der Wert nach Aufruf der Funktion Null, so war die Ausführung erfolgreich.

9.7. Das Testprogramm PCTest2

Dieses Programm demonstriert die grundlegende Vorgehensweise bei der Programmierung von Bewegungsabläufen mit den PowerCube™-Funktionen. Der Ablauf ist:



9.8. Umrechnung von Inkrementen in Grad bzw. Millimeter

Die Umrechnung von Inkrementen in gebräuchliche Einheiten, wie Millimeter oder Grad erfolgt anhand der im Produktbegleitbrief angegebenen Daten:

- Getriebeübersetzung (GearRatio): gera
- Inkremente pro Motorumdrehung (PulsesPerTurn): ppt
- Durchmesser der Zahnriemenscheibe (ToothWheelDiameter), nur für Zahnriemenachsen: twd
- Spindelsteigung (Lead), nur für Spindelachsen und Greifer: ld

Antriebsart	SI-Einheit => Inkremente	Inkremente => SI-Einheit
Drehantriebe	ink = Grad*ppt*4*gera/360	Grad = ink*360/(ppt*4*gera)
Zahnriemenantriebe	ink = mm*ppt*4*gera/(twd*p)	mm = ink*twd*p/(ppt*4*gera)
Spindelantriebe und Greifer	ink = mm*ppt*4*gera/ld	mm = ink*ld/(ppt*4*gera)

Tabelle 130

mm: Millimeter / Grad: °